TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: Desenvolupament d'una aplicació de planificació de vol i de comunicació entre simuladors de vol

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica

AUTOR: Iván Lorente Díez

DIRECTOR: Xavier Prats i Menéndez

DATA: 24 de febrer de 2005

Títol: Desenvolupament d'una aplicació de planificació de vol i de comunicació entre simuladors de vol

Titulació: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica

Autor: Iván Lorente Díez

Director: Xavier Prats i Menéndez

Data: 24 de febrer de 2005

Resum

Els objectius principals d'aquest projecte són el desenvolupament d'una aplicació client/servidor i el seu propi protocol de comunicacions per gestionar una xarxa de simuladors de vol i una aplicació capaç de generar informació de meteorologia aeronàutica (METARs). L'aplicació client/servidor incorpora un mòdul de planificació del vol pels pilots capaç de consultar aquesta informació i generar i enviar plans de vols normalitzats al servidor.

Les dues aplicacions tenen com a finalitat el seu ús a les classes pràctiques de diferents assignatures de la titulació d'Enginyeria Tècnica Aeronàutica esp. en Sistemes d'Aeronavegació que s'imparteix a l'Escola Politècnica Superior de Castelldefels.

També s'ha fet un estudi de les diferents eines existents que realitzen funcions similars a les proposades. Aquest estudi és el punt de partida i de referència per a definir les característiques i funcionalitats que tindran les nostres aplicacions.

Title: A flight planner and communication between flight simulators application development

Degree: Technical Telecommunications Engineering, Telematics speciality

Author: Iván Lorente Díez

Director: Xavier Prats i Menéndez

Date: February, 24th 2005

Overview

The main objectives of this project are the development of an application client/server and its own communications protocol to manage a flight simulator network, and the development of an application that generates aeronautical meteorological information reports (metar). The client/server application has a flight planner module to check this meteorological information and other information and send standardized flight plans to the server.

These two applications have the purpose to be used at the practical classes of the Technical Aeronautical Engineering, Aero Navigation Systems speciality degree, which is cursed at the Escola Politècnica Superior de Castelldefels.

Finally it's been made a study of the actual software we can find currently that works similar at described above. This study works as a reference to design and develop the main characteristics and functionalities of our own applications.

ÍNDEX

INTRODUCCIÓ

Idea i raó del projecte

Des de fa 3 anys, l'Escola Politècnica Superior de Castelldefels imparteix la titulació d'Enginyeria Tècnica d'Aeronàutica esp. en Sistemes d'Aeronavegació. Algunes de les assignatures cursades en aquesta carrera utilitzen simuladors de vol, algunes vegades connectats en xarxa, per tal de mostrar de manera pràctica als alumnes aspectes de pilotatge, gestió de l'espai aeri, navegació, tecnologia aeronàutica, etc...

La utilització de les aplicacions que hi ha al mercat, moltes d'elles gratuïtes però de codi tancat, impedeixen una gestió personalitzada d'aquestes xarxes de simulació.

Objectius

Es planteja la idea de desenvolupar un protocol de comunicacions genèric, que es pugui utilitzar amb independència del simuladors de vol que es vulgui utilitzar. D'aquesta manera només s'haurà d'implementar una petita aplicació, diferent a cada simulador de vol, capaç de comunicar les dades necessàries entre ell mateix i el client de la nostra aplicació. La creació d'aquests *traductors* entre el simulador de vol i el client no és objectiu d'aquest projecte, ja que s'espera que futurs estudiants continuïn amb la tasca començada i siguin capaços d'implementar aquestes aplicacions

A la mateixa vegada sorgeix la idea de crear una aplicació capaç de generar informació meteorològica aeronàutica, *metar*, amb el seu format aeronàutic estàndard, una vegada vista la manca d'aplicacions d'aquestes característiques al mercat, i adequat a les característiques meteorològiques espanyoles. Cal dir, però, que seguint els esquemes realitzats és molt senzilla la modificació dels paràmetres introduïts per tal d'adequar aquesta aplicació a diferents entorns meteorològics.

Estructura de la memòria

Al capítol 1 es presenten les necessitats de les comunicacions aeronàutiques, qui les regula i quins elements hi intervenen.

Al capítol 2 hi ha una presentació de software existent que fa les funcions de xarxa de simulació de vol. En aquest mateix capítol es definiran quines són les necessitats de la nostra xarxa de simulació de vol: usuaris, informació a transmetre i aplicacions necessàries per a crear la xarxa de simulació.

Al capítol 3 es dissenyen les aplicacions client/servidor i el protocol creat per a la seva comunicació.

Al capítol 4 hi ha una presentació sobre la informació meteorològica aeronàutica i com s'estructuren els informes.

Al capítol 5 trobem el disseny de l'aplicació capaç de generar aquests informes d'informació meteorològica aeronàutica.

Al capítol 6 trobem les conclusions i balanços.

1. TELECOMUNICACIONS AERONÀUTIQUES

1.1. Necessitat de les telecomunicacions en aeronàutica

El 1903 els germans Wright van aconseguir enlairar 10 metres d'alçada el primer avió propulsat amb motor durant 250 metres. Aquell dia va néixer el que es coneix com a *Aviació*.

Ja des de llavors, es va observar la necessitat d'establir enllaços de comunicacions aire-terra. Les telecomunicacions aeronàutiques han esdevingut una de les principals missions dels organismes encarregats de la regulació aeronàutica.

Els objectius principals dels Serveis de Trànsit Aeri, són la seguretat, l'ordre, la fluïdesa, i l'economia, en aquest ordre, i en un vol típic, les aeronaus mantenen comunicació amb diferents elements d'aquests Serveis per tal d'assolir els objectius.

1.2. Organismes i legislació de les Telecomunicacions Aeronàutiques

Al 1919, al finalitzar la 1a Guerra Mundial, es va crear la Conferencia Internacional de Navegació Aèria (CINA) per controlar el tràfic aeri, que va crear per primera vegada una regulació aeronàutica.

Però no és fins al final de la 2a Guerra Mundial, quan es reuneixen la majoria de països per tal d'establir un control aeri general. Neix doncs, la OACI, la Organització d'Aviació Civil Internacional, fruit del conveni de Chicago de 1944, per a l'aplicació d'aquest conveni i completar-lo amb els Annexos Tècnics.

Són molts els organismes relacionats amb les diferents regulacions aèries, tant nacionals (DGAC –Direcció General d'Aviació Civil– , CIDETRA, CIPAI, SCA, AECA, ALA, AOC) com internacionals (OACI, CEAC, EUROCONTROL, JAA, CANSO, IATA...).

La OACI, però, va realitzar una sèrie d'Annexos al Conveni de Chicago per tal de regular oficialment la majoria dels aspectes. La majoria de països del món segueixen aquesta normativa internacional.

És a l'Annex 10 del Conveni de Chicago on s'estableix la normativa de Telecomunicacions Aeronàutiques, que reflecteix aspectes tant diferents com els espectres de freqüència de les ràdios, procediments generals, serveis de radionavegació, categoria de missatges, prioritats dels missatges, etc.

1.3. Tipus de telecomunicacions aeronàutiques

Actualment i des de fa anys, en un vol típic, una aeronau manté enllaços de comunicació amb diferents elements per tal de poder assolir els objectius dels Serveis de Trànsit Aeri. Tot seguit es descriuen breument les principals aplicacions de les telecomunicacions aeronàutiques. Els detalls d'aquestes són descrits a l'Annex 10 del conveni de Chicago (veure [5]).

1.3.1. Sistemes de comunicacions

Una aeronau necessita un sistema de comunicacions amb els serveis de trànsit aeri per tal de poder facilitar les tasques de vigilància i així evitar col·lisions amb altres aeronaus i per poder demanar i rebre les autoritzacions necessàries per tal d'operar en zones on l'espai aeri està regulat. A més a més pot beneficiar-se d'aquestes comunicacions per demanar informació que li pugui resultar d'interès (meteorologia, navegació, ...) i utilitzar-les en cas d'emergència per alertar els serveis de cerca i salvament.

Tot això es materialitza amb comunicacions ràdio de veu a través de canals VHF (HF per llargues distàncies). També existeixen enllaços digitals de comunicacions entre pilots i controladors aeris per tal de optimitzar les comunicacions, tot i que avui en dia aquestes tècniques estan en fase d'implantació.

1.3.2. Sistemes de navegació

Els sistemes de navegació són aquells equips electrònics instal·lats a la superfície terrestre o en òrbita a l'espai, capaços d'indicar a les aeronaus quina és la seva localització a la superfície terrestre o d'ajudar-les a deduir-la gràcies a la informació que proporcionen.

Alguns sistemes de navegació són molt coneguts, com el GPS (o Sistema de Posicionament Global) que utilitza satèl·lits en òrbita per determinar la posició de l'aeronau, o sistemes més convencionals com els ADF, els VOR i els DME que permeten, gràcies a la seva informació i a la interpretació de cartes aeronàutiques deduir la posició de l'aeronau.

Fig. 1.1 Captura de pantalla de l'indicador ADF d'una Cessna

1.3.3. Sistemes de vigilància

La vigilància aèria avui en dia està assegurada per radars primaris (radar convencional) i secundaris (l'avió activament envia la seva identificació quan és detectat pel radar primari). En zones oceàniques, on no hi ha cobertura radar, els avions són controlats a partir d'informes de comunicacions per veu dels mateixos pilots, limitant considerablement la capacitat de l'espai aeri.

1.3.4. Els ATC

Els ATC (air traffic control), o sigui els controladors de trànsit aeri, són aquell element del conjunt que s'encarrega de garantir la separació d'aeronaus per tal de assegurar una operació segura i eficient d'aquestes. Utilitza els sistemes de vigilància per monitorizar les aeronaus i els sistemes de comunicacions per poder donar les informacions o ordres pertinents. La funció dels controladors de tràfic aeri i dels Serveis de Trànsit Aeri és imprescindible en zones on el volum de tràfic aeri és alt.

Existeixen diferents tipus de controladors de trànsit aeri, i les principals diferències resideixen a l'espai aeri que controlen. Així, una aeronau es posarà en contacte amb un determinat controlador aeri depenent de la fase de vol on estigui. Si per exemple la fase és la d'enlairament, l'aeronau s'ha de posar en contacte amb els controladors de torre de l'aeroport que portin les sortides. Una vegada enlairat connectarà amb els controladors de ruta.

Fig. 1.2 Exemple dels llocs de treballs de diferents ATC

1.4. Informació i dades transmeses

1.4.1. Aeronau – ATC

La informació transmesa entre una aeronau i un ATC pot ser, generalment, de dos tipus:

Veu: Tota aquella informació que es comuniquen pilots i controladors mitjançant radio. Normalment són ordres i confirmacions.

Dades: Aquella informació que l'aeronau està enviant contínuament de manera transparent per als pilots o que està sent rebotada a l'aeronau (sistema radar), però que queda reflectida a les pantalles i els sistemes informàtics dels controladors aeris.

Les dades que es visualitzen als centres de control poden ser enviades per les aeronaus, aconseguides amb sistemes radar, i calculades pels mateixos sistemes de control. Són les següents:

- *Transponder*: Identificador de l'aeronau. Enviada per les aeronaus.
- *Altitud*: Altitud indicada de l'aeronau.
- *Posició*: Els sistemes radar determinen la posició de les aeronaus.
- *Velocitat*: Els sistemes de control calculen les velocitats de les aeronaus mitjançant les posicions en cada moment.
- *Ruta*: Els sistemes de control calculen la ruta que, presumiblement, farà una determinada aeronau mitjançant les posicions, velocitats, pla de vol presentat, etc.

1.4.2. Aeronau – Aeronau

Algunes aeronaus de nova construcció incorporen radars als seus equips electrònics per tal d'augmentar la seguretat i poder detectar les altres aeronaus.

Els pilots de diferents aeronaus poden comunicar-se via ràdio amb d'altres col·legues, però no és habitual.

1.4.3. Aeronau - Sistemes de navegació

Les aeronaus i els diferents sistemes de navegació instal·lats pels Serveis de Trànsit Aeri, tan terrestres com satel·litars, es comuniquen mitjançant les tècniques documentades a l'Annex 10 del Conveni de Chicago.

De vegades la comunicació esdevé unidireccional o bidireccional. Això és degut a que determinats sistemes de navegació difonen uns missatges per l'entorn, i són les aeronaus que, una vegada els reben són capaços d'interpretar-los i analitzar-los per tal de determinar la seva posició respecte el focus emissor dels missatges. És el cas dels sistemes ADF.

Un cas de bidireccionalitat seria l'utilitzat en sistemes DME (equips mesuradors de distància). L'aeronau envia un missatge, que rep un DME i contesta. Calculant el temps que triga la resposta són capaços d'esbrinar la distància a la que es troba l'aeronau d'aquest DME.

2. ANÀLISI DEL SOFTWARE EXISTENT I SOLUCIÓ PROPOSADA

2.1. Software escollit

Existeixen moltíssimes aplicacions destinades a la simulació aèria. A Internet podem trobar molts fans i grans organitzacions dedicades a la simulació de trànsit aeri. Moltes d'aquestes organitzacions i els seus membres, es dediquen al desenvolupament de software dedicat a la seva afició i són molts les aplicacions dedicades a tal efecte.

Algunes d'aquestes comunitats virtuals, creen les seves pròpies xarxes de simulació i les seves pròpies aplicacions. Una d'elles és IVAO, la International Virtual Aviation Organization, que compta amb més de 38000 usuaris registrats de tot el món i és, probablement una de les organitzacions de simulació de vol més fortes del món.

Les xarxes virtuals de tràfic aeri funciones amb arquitectures client-servidor, on els clients poden ser pilots o controladors aeris, i existeix un servidor central que s'encarrega de gestionar, emmagatzemar i redireccionar la informació cap als clients corresponents.

Tot seguit presentaré un petit anàlisi d'algunes de les aplicacions més utilitzades en l'àmbit de les simulacions de tràfic aeri. Només s'han tingut en compte aquelles aplicacions en que la seva utilització és orientada a una LAN de caràcter privat. Així doncs, excloem aquelles aplicacions creades exclusivament per a determinades xarxes i comunitats, com podrien ser les aplicacions IVAX (tant clients pilot com clients ATC) desenvolupades exclusivament per a connectar-se a servidor de la organització.

2.1.1. Servidor: FSD

El servidor FSD és possiblement un dels més utilitzats per a gestionar xarxes virtuals de tràfic aeri de caràcter públic però també privat. No en va, era i és actualment l'utilitzat a vàries de les assignatures de la titulació Enginyeria Tècnica d'Aeronàutica esp. en Sistemes d'Aeronavegació.

Entorn: És una aplicació WIN32 i com tal només permet el seu funcionament en entorns Windows. Les últimes versions d'aquest servidor estan sortint també per a sistemes basats en UNIX.

Interfície d'usuari: No té una interfície directa amb l'usuari: una vegada activat s'ha de fer un telnet al port 3012 per a poder accedir a les opcions de gestió. A més, aquesta interfície (en mode text) és bastant rudimentària i no és gens agradable. Les opcions són molt confuses.

Fitxers que necessita: El servidor necessita diversos fitxers per al seu funcionament. Existeix un fitxer de configuració (fsd.conf) on es poden

modificar diferents opcions tal com el port de connexió amb els clients, el password per a accedir a les opcions del server, el nom del fitxer de certificats, etc.). El fitxer de certificacions és un fitxer .txt on hi ha una relació dels usuaris que poden accedir al servidor, amb el seu password i la seva categoria.

Funcionament intern: Desconeixem el funcionament intern de l'aplicació, ja que, encara ser un software de caràcter gratuït, no ho és de codi obert. S'està realitzant un projecte per redissenyar aquest servidor i fer-lo de codi obert.
El servidor és capaç de gestionar les connexions de pilots i controladors aeris, així com la informació que s'envien automàticament els uns als altres.

2.1.2. Client pilot: SQUAWKBOX

L'Squawkbox és una de les aplicacions més importants a l'entorn dels clients pilot. Es va dissenyar per a la seva utilització com a add-on per al Microsoft Flight Simulator (s'han anat actualitzant les versions per a cobrir la majoria d'ells). Ara però, podem trobar modificacions per a utilitzar-lo en altres simuladors de vol com ara l'X-Plane o el FlightSim.

Entorn: Tal i com s'acaba de comentar, l'Squawkbox està especialment dissenyat per el seu funcionament conjunt amb el MS Flight Simulator, que corre sobre entorns Windows. Tanmateix, les altres possibilitats d'utilització en d'altres simuladors de vol també s'encaren a la utilització en entorns Windows.
Només es deixa instal·lar si es té el simulador de vol necessari instal·lat prèviament.

Interfície d'usuari: La interfície gràfica és un tant estranya, ja que utilitza un menú desplegable per a seleccionar les opcions, en comptes de les finestretes de Windows que farien l'aplicació més amigable.

Fig. 2.1 Captura de pantalla de l'Squawkbox

Funcionament intern: Aquest també és, de moment, una aplicació gratuïta però de codi no obert. Aquest client es connecta a un FSD Server (explica't a l'apartat anterior). Una vegada connectat, es poden utilitzar les diferents funcionalitats del client, especialment indicades a l'apartat de planificació de vol, com ara l'enviament del pla de vol. Pot visualitzar els ATC connectats al servidor i té la possibilitat de seleccionar amb quin d'ells es vol comunicar. Mitjançant un software de veu com ara Roger Wilco, de poder utilitzar la veu per tal de comunicar-se amb els ATC (després d'introduir la freqüència determinada d'un ATC).

2.1.3. Client ATC: PRO CONTROLLER

Entorn: Com l'Squawkbox, el Pro Controller està especialment dissenyat per el seu funcionament conjunt amb el MS Flight Simulator en entorns Windows.

Interfície d'usuari: La interfície gràfica és més amigable que la de l'Squawkbox. Es presenta en una finestra dividida en 4 parts. A cadascuna d'elles es reflexa una funcionalitat. A la divisió principal trobem una simulació d'una pantalla d'un ATC on, una vegada connectats al FSD Server, podem veure les aeronaus i els seus identificadors. En una altra trobem la freqüència a la que opera el ATC i a la que s'han de connectar els pilots.

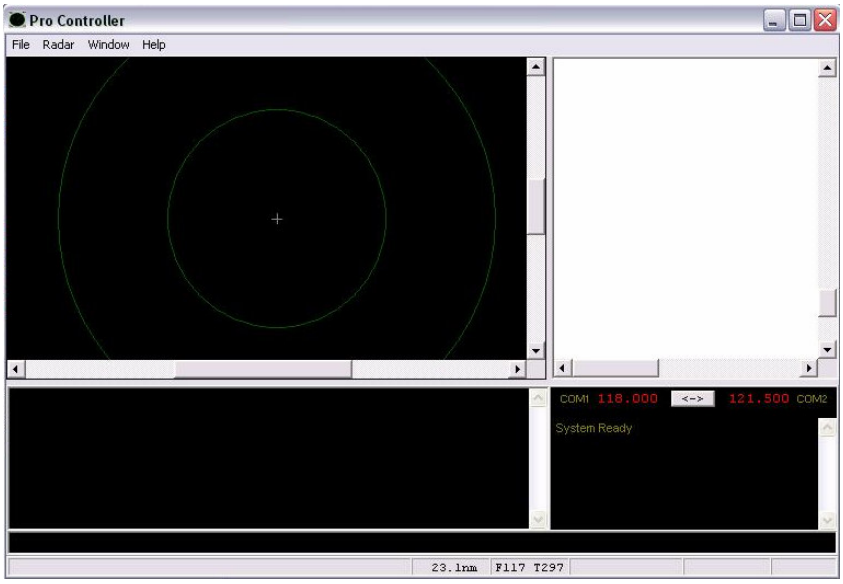


Fig. 2.2 Captura de pantalla de el Pro Controller

Fitxers que necessita: Se li poden afegir *sector-files*, fitxers que determinen sectors aeris determinats, per tal de que la pantalla de control sigui més realista.

Funcionament intern: El Pro Controller, com la resta de les aplicacions escollides, és gratuït però de codi tancat. Aquest programa ens permet

visualitzar en un sector aeri determinat la situació real de cada aeronau. Permet múltiples canals de comunicació amb els pilots i altres ATC. També permet la comunicació de veu mitjançant un plug-in adient.

2.1.4. Meteorologia: FSMETAR

Entorn: Windows amb el Microsoft Flight Simulator instal·lat i el FSUIPC (aplicació capaç de extreure i introduir dades al MSFS).

Funcionament intern: Aquesta aplicació extreu mitjançant l'FSUIPC la posició de l'aeronau i descarrega automàticament el metar real més proper a la posició indicada i introdueix les dades meteorològiques rebudes a través també de l'FSUIPC.

2.2. Necessitats de la xarxa virtual de simulació de vol

La normativa i la informació transmesa esmentada al primer capítol són aplicables al tràfic real. La nostra aplicació, però, és una simulació informàtica i serem nosaltres els que programarem la nostra *pròpia normativa* o protocol de comunicació, i quines són les dades a transmetre.

En aquest apartat ens centrarem en quins són els usuaris que haurien d'utilitzar la xarxa de simulació de vol i quines són les dades que s'han de transmetre els uns als altres.

2.2.1. Usuaris

Hi ha 2 tipus de clients, els clients pilots i els clients ATC. Així doncs, la nostra xarxa serà molt semblant a la que es crea mitjançant el software analitzat anteriorment, però amb la diferència de com s'envien la informació (ho veurem al capítol de disseny de l'aplicació) i de quina informació volem enviar i/o rebre.

2.2.2. Informació a transmetre

En aquest projecte ens centrarem en la informació a transmetre entre els avions i els Serveis de Trànsit Aeri i especialment en el Control de Trànsit Aeri (ATC), i no en la informació transmesa entre les aeronaus i els sistemes de navegació, ja que això ho fan els mateixos simuladors de vol.

Les dades a enviar, rebre i gestionar són:

- *Login*: Ha de ser capaç de donar accés als usuaris. Ha de validar un nick i contrasenya. Transparentment a l'usuari i depenent de l'aplicació client, ha de saber si es tracta d'un pilot o d'un ATC.
- *Metar*: El servidor ha de ser capaç de servir els metars dels diferents aeroports als clients pilots que els sol·licitin. Aquests metars els ha de

crear ell mateix mitjançant l'aplicació Metar dissenyada a capítols posteriors.
- *Notam*: El servidor ha de ser capaç de servir els notams dels diferents aeroports en uns determinats rangs de dates i hores. Els notams són introduïts per l'administrador amb un fitxer de text a la mateixa màquina on es troba el servidor.
- *Pla de vol*: Els clients pilot han d'enviar el pla de vol abans de començar el vol i els clients ATC l'han de rebre, acceptar-lo o denegar-lo i informar al pilot la seva decisió.
- *Posicions*: Els clients pilot han d'enviar, de manera transparent als usuaris de l'aplicació, la posició de la seva aeronau als ATC per a que aquests els puguin controlar.

2.3. Solució proposada

La característica principal del software escollit és que és gratuït però de codi tancat, i per tant no podem modificar-lo per assolir les nostres necessitats. També trobem la postura contrària: té possibilitats que no ens interessen, i no les podem eliminar.

La idea és realitzar una xarxa semblant a la creada pel software escollit però amb la capacitat de poder ampliar les seves funcionalitats i adequar-les a les nostres.

Tot seguit es presenta la solució proposada i les aplicacions que es necessiten per a crear una xarxa d'aquest tipus.

2.3.1. Servidor

Es necessita un servidor per tal de gestionar els usuaris i la transferència de dades i informació.

Aquest servidor s'implementarà en un entorn UNIX, per tal de garantir el seu bon funcionament, i per la capacitat d'atendre a diferents clients de manera simultània. S'utilitzarà el llenguatge C de UNIX per a programar l'aplicació.

Ha de ser capaç de servir i rebre les dades, explicades al capítol anterior, que li demanin/enviïn els clients. El protocol utilitzat s'explicarà al capítol del disseny del servidor.

L'administrador de la xarxa de simulació de vol ha de poder demanar algunes dades al servidor, com el número de clients connectats (ATCs i/o pilots), hora local i hora UTC, un metar determinat, tancar el servidor, etc...

2.3.2. Client pilot

És una aplicació client capaç de demanar i enviar les dades necessàries per a la planificació del vol.

Com que la majoria de simuladors de vol treballen en entorns Windows, el client es programarà en aquest entorn i en sistema visual.

Han d'enviar el nom d'usuari i password per a accedir a la xarxa, poder demanar metars i notams dins d'un rang temporal. Ha de poder rebre les dades de posició per part del simulador de vol, adaptar-les per a que les entengui el servidor i enviar-les al servidor. Ha de tenir un apartat per a poder omplir un pla de vol i enviar-lo, i saber si aquest pla de vol ha estat acceptat o denegat per un ATC (si ha estat denegat s'ha de poder veure perquè).

2.3.3. Client ATC

En aquest apartat s'explicarà com, idealment, hauria de funcionar un client ATC, però la seva implementació no és un dels objectius d'aquest projecte.

Aquest client ha de poder rebre i enviar dades per tal de gestionar la simulació de vol, així com acceptar o denegar plans de vol, ajudar als clients pilot, etc.

Un client ATC ha de rebre les posicions de les aeronaus i poder veure-les gràficament a una pantalla. A més ha de ser capaç de rebre un pla de vol normalitzat i enviat per un pilot, visualitzar-lo i l'usuari ha de poder acceptar-lo o denegar-lo segons cregui convenient. Si s'ha denegat, s'ha d'enviar un missatge indicant perquè paràmetres s'ha denegat.

2.3.4. Meteorologia

En comptes d'un sistema que descarrega la informació meteorològica d'Internet, es dissenyarà una aplicació capaç de generar la seva pròpia informació aeronàutica, per no haver de dependre d'una connexió a Internet. A més, l'administrador de la xarxa de simulació de vol, podria modificar certs paràmetres de l'aplicació per tal de simular una meteorologia més o menys concreta.

Aquesta aplicació funcionarà conjuntament amb el servidor i per tant s'implementarà també en C UNIX. El servidor l'executarà cada un cert temps (per defecte 30 minuts) per tal d'actualitzar els metars com a la vida real.

Els metars han d'adaptar-se i ser congruents amb l'època de l'any i la hora en que s'està executant el programa per tal de ser el més realista possibles.

3. DISSENY DE L'APLICACIÓ *CLIENT/SERVIDOR*

3.1. Descripció de l'aplicació

Durant la redacció d'aquesta part del projecte es va plantejar la idea de redactar el disseny de l'aplicació en 2 capítols, un per al client i un altre per al servidor. Degut però a que és molt difícil explicar el funcionament de només una part d'una arquitectura client/servidor, s'ha optat per fer una redacció conjunta dels dos elements per tal de fer-lo més entenedor.

3.1.1. Objectiu de l'aplicació

L'objectiu principal és la implementació d'una sèrie d'aplicacions per tal de crear una xarxa virtual de simulació de vol. Aquesta xarxa té com a finalitat la realització de simulacions de vol per part dels alumnes de la titulació Enginyeria Tècnica d'Aeronàutica, especialitat en Sistemes d'Aeronavegació, que realitzen aquest tipus de pràctiques a diferents assignatures de la titulació.

La idea és la d'interconnectar diferents pilots i controladors aeris i integrar-los en un mateix àmbit per tal d'aproximar la simulació el més possible a la vida real.

3.1.2. Arquitectura utilitzada

Com la majoria d'aquest tipus de xarxes, s'utilitzarà una arquitectura client/servidor per tal de gestionar tota la informació. Com ja hem vist al capítol anterior, s'implementaran:

- *Servidor*: S'encarrega de gestionar la informació, emmagatzemar-la i servir-la als clients que li demanin.
- *Client pilot*: Demana informació variada (metars, notams, etc...) al servidor, rep les respostes i les mostra per pantalla.
- *Aplicació generadora de metars*: Genera informes metar. Aquesta aplicació es dissenyarà al següent capítol.

En aquest capítol ens centrarem en el disseny, implementació i funcionalitats de les aplicacions servidor i client pilot.

3.1.3. Entorn de treball

Com ja s'ha esmentat a la solució proposada del capítol 2, el servidor s'implementarà amb C UNIX, degut a les seves bones prestacions a l'hora de realitzar servidors, ja que permet de manera bastant senzilla la creació de processos fills per tal d'atendre les diferents peticions del client, i a la facilitat del llenguatge.

Respecte als clients, el client pilot s'implementarà amb Visual Basic, ja que la majoria de simuladors de vol treballen en entorn Windows i es podran executar a la mateixa màquina tan simulador de vol com client de connexió. A més la interfície gràfica de l'usuari és molt amigable i les funcionalitats del sistema són molt fàcils d'utilitzar en entorns Windows. Es podien haver implementat les funcions amb C++ i només la interfície gràfica amb Visual Basic, però es va desestimar degut a que aquestes no eren extremadament complicades i seria més senzill fer-ho directament en Basic que no pas després integrar les funcions en C++ amb l'interfície gràfica.

3.1.4. Diagrama de flux del servidor

Tot seguit es presenta el diagrama de flux del servidor implementat, per tal de veure el funcionament intern d'una manera esquemàtica.

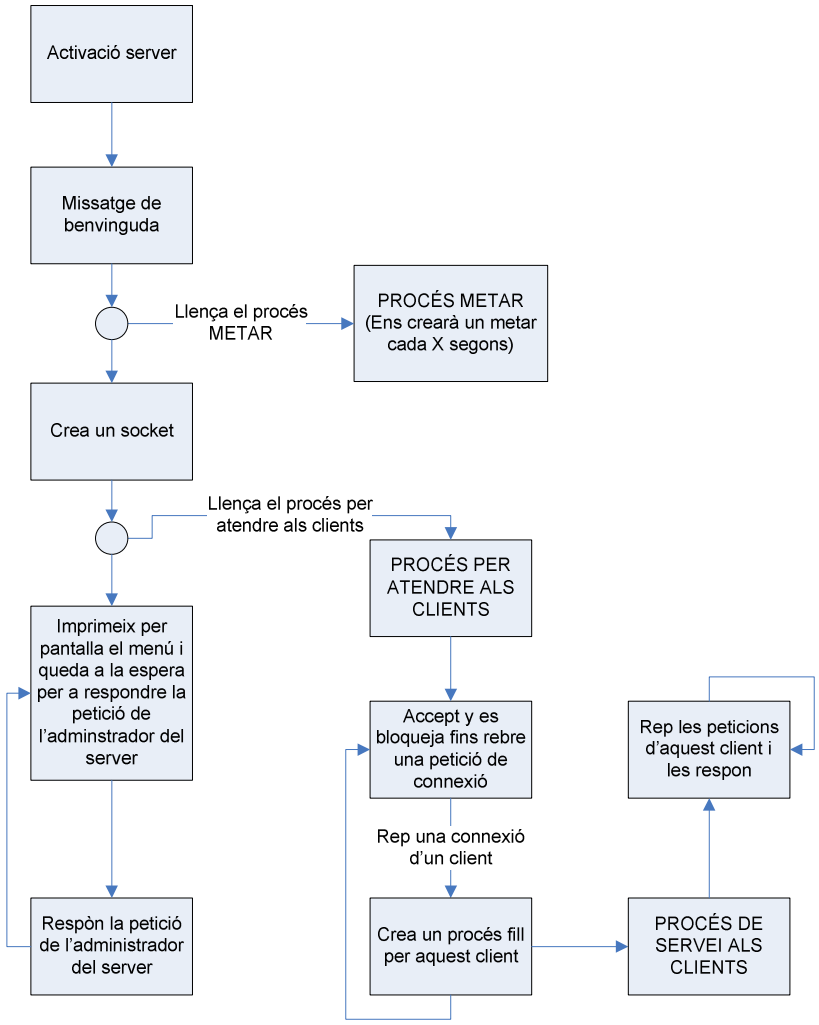


Fig. 3.1 Esquema servidor

3.2. Descripció de fitxers utilitzats i interfície gràfica

En aquest apartat es descriu l'estructura de cadascuna de les aplicacions, i els fitxers utilitzats en la mateixa.

3.2.1. Servidor

L'aplicació servidor ha estat dissenyada amb C UNIX, i el seu funcionament s'ha dividit en diversos arxius per tal de modularitzar la seva programació.

3.2.1.1. Server.c

El fitxer principal és *server.c*, on es defineix el servidor i les crides a la resta de funcions. En aquest arxiu es defineixen principalment la funció main del programa i les funcions més específiques de servidor.

La funció main de l'aplicació és bastant senzilla gràcies a aquesta modularització. S'encarrega en aquest ordre de:

1. Gestionar els arguments a l'invocació
2. Programar les signals
3. Imprimir per pantalla el missatge de benvinguda
4. Llençar el procés generador d'informació meteorològica
5. Crear un socket d'escolta
6. Cridar a la funció que crearà un procés per atendre als clients i passar-li el descriptor del socket per atendre als clients
7. Imprimir per pantalla el menú del servidor i la seva gestió

En aquest mateix fitxer trobem la programació de les signals, i les funcions que creen el socket, que també s'han modularitzat.

3.2.1.2. Funciones_server.c

En aquest fitxer hi ha la majoria de les funcions a les que crida el servidor per tal d'assolir els seus objectius. Entre elles tenim la funció que crea un procés fill per a atendre els clients, o la que crea un altre per llençar el procés generador de metars. Aquests funcions les veurem amb més detall al l'apartat 3.4.1, destinat a la descripció de les funcions utilitzades pel servidor.

3.2.1.3. Estructuras.h

En aquest fitxer s'han definit les estructures en C necessàries per a la gestió de la informació que s'envia i/o es rep a/des de el servidor. La seva definició i els tamanys dels strings coincideixen perfectament amb les descrites a l'apartat 3.3 d'aquest mateix capítol, dedicat precisament a les estructures de dades. Com

s'explica a l'apartat esmentat, el tamany de les estructures definides en Visual Basic és a les funcions de mapeig de les estructures en strings per a ser enviades pel socket.

3.2.1.4. info_server.h

En aquest fitxer s'han definit diversos paràmetres del servidor com ara l'adreça IP de la màquina que allotja el servidor, el port de connexió i el número màxim de connexions clients simultànies. Aquest fitxer no és realment necessari ja que l'adreça IP del servidor i el número de port es passen directament al servidor a la invocació del mateix i el número màxim de connexions pot estar definit al fitxer *server.c* amb un define. Però si el servidor no ha de canviar de màquina i sempre restarà a la mateixa, amb la mateixa adreça IP, és bona idea no haver d'escriure cada vegada que el volem activar l'adreça i el port.

3.2.1.5. Pass.txt, Notam.txt i fitxers de metars

En aquests fitxers es troba la majoria d'informació que es servirà als clients.

El fitxer **pass.txt** conté una sèrie de línies amb els noms d'usuari, contrasenya i tipus d'usuari que poden accedir a la xarxa. L'estructura del fitxer és la següent:

Usuari1_pass1_Tipus
Usuari2_pass2_Tipus

on Tipus és *A* si és un ATC o *P* si és un Pilot.

El client ha d'enviar aquests dades al servidor i aquest les compara amb les que té a la llista. Si troba una cadena idèntica a la del fitxer, valida a l'usuari.

Notam.txt és el fitxer on l'administrador del sistema afegeix els Notams segons cregui convenient. L'estructura del fitxer de notams és la següent:

AERO DATA1 HORA1 DATA2 HORA2 NOTAM#

on:

AERO: Codi OACI de l'aeroport
DATA1: Data d'inici de validesa del notam
HORA1: Hora d'inici de validesa del notam
DATA2: Data de finalització de validesa del notam
HORA2: Hora de finalització de validesa del notam
NOTAM: És la informació pròpiament dita
#: indica al servidor que el notam acaba aquí

Els **fitxers de metars** corresponen a aquells que contenen els informes meteorològics aeronàutics d'un determinat aeroport. Del seu funcionament, i la seva generació s'en parlarà extensament als capítols 4 i 5 d'aquest projecte.

3.2.2. Client Pilot

Aquesta aplicació s'ha dissenyat amb Visual Basic, i per tant s'ha creat un projecte. Els fitxers que s'han utilitzat corresponen als formularis que l'integren, més un mòdul que s'ha utilitzat per desenvolupar les funcions públiques i definir les estructures de dades.

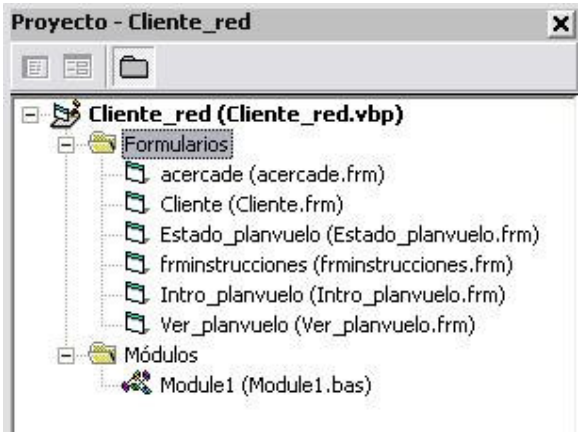


Fig. 3.2 Fitxers que s'inclouen al projecte. Aplicació client pilot

Tot seguit s'explicaran els dos formularis més importants i les seves funcions.

3.2.2.1. Cliente.frm

El formulari *Cliente* és el principal i des d'on s'executen la majoria de les funcionalitats de l'aplicació. Des d'aquest mateix formulari es realitza la connexió amb el servidor, es demanen els notams i metars, i s'envien les posicions de l'aeronau (això ho deuria de fer implícitament i de manera transparent per a l'usuari, però per comprovar la seva funcionalitat les posicions s'hauran d'enviar manualment).

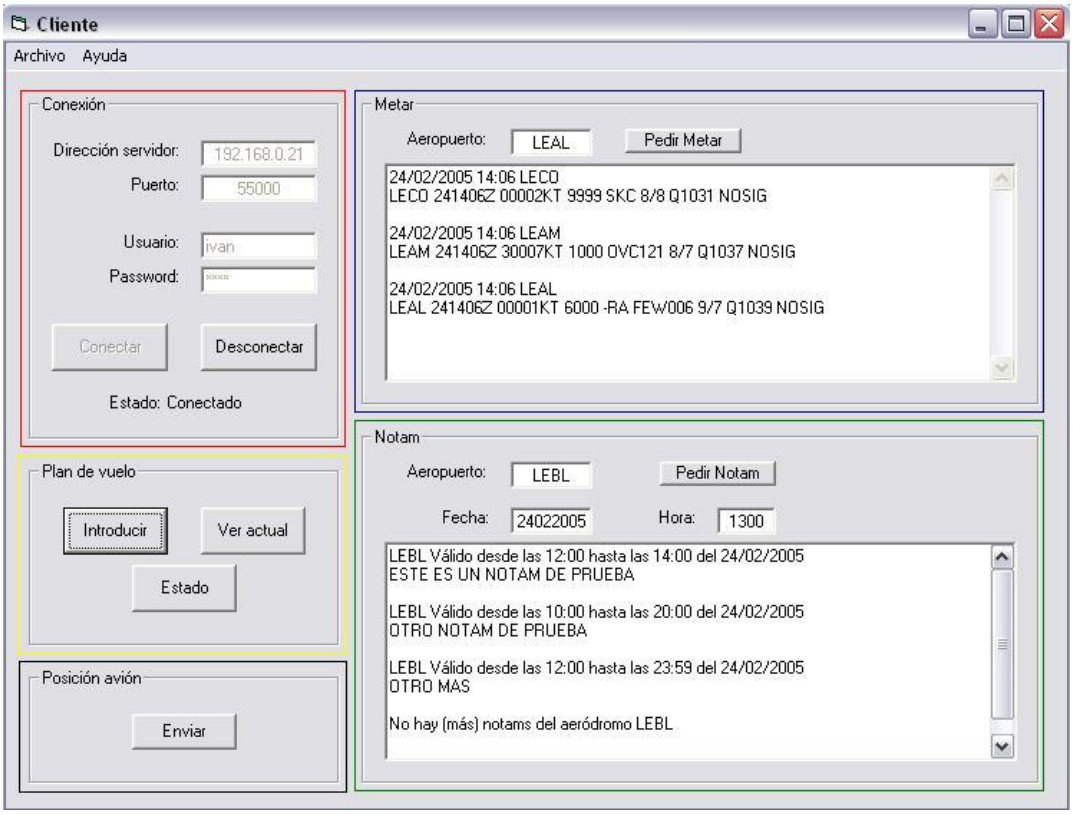


Fig. 3.3 Formulari *Cliente.frm*

A la figura es poden veure els diferents apartats i funcionalitats de que disposa aquest formulari.

En vermell trobem l'apartat de connexió. En aquest apartat s'ha d'introduir tan les dades de connexió, adreça IP del servidor i port de connexió, com les dades de l'usuari per tal de validar-lo. En el mateix requadre podem observar els controls Winsock per definir sockets en Windows. El primer és per la connexió TCP amb el servidor, i el segon, per la connexió UDP amb el simulador de vol X-Plane.

Inicialment, fins que no s'estableix la connexió i es valida a l'usuari, tota la resta de funcionalitats no s'activen.

En blau hi ha l'apartat de la demanda de metars. L'usuari ha d'introduir el nom de l'aeroport en format OACI del qual vol veure el metar, i clickar el botó "Pedir metar". Tot seguit, el servidor retorna el metar concret i el mostra a la finestra corresponent.

En verd trobem l'apartat notam, molt semblant a l'anterior. S'ha d'introduir el nom de l'aeroport, i la data i l'hora del qual es vol saber si hi ha notams. El servidor retorna una llista dels notams que hi ha publicats i els mostra per la pantalla.

En groc hi ha l'apartat del Pla de Vol. Existeixen 3 botons. El d'introduir ens mostra el formulari *Intro_planvuelo* per introduir les dades del Pla de Vol normalitzat, guardar-lo en un fitxer de l'estil *pla_de_vol.pv* (per a poder tornar-lo a obrir amb *Ver_planvuelo*) i enviar-lo al servidor. El botó *Estado* ens obre el formulari que ens indica si el pla de vol ha estat acceptat o denegat per part del controlador aeri que l'ha revisat. Si ha estat denegat, en aquest mateix formulari hi ha explicats els motius.

En negre trobem el botó per enviar l'última posició rebuda des del simulador de vol.

3.2.2.2. Intro_planvuelo.frm

A la figura es pot observar el formulari per introduir el Pla de Vol normalitzat. Cal dir que aquest formulari s'ha creat basant-se en el model oficial de la OACI, i és el model real que s'utilitza.

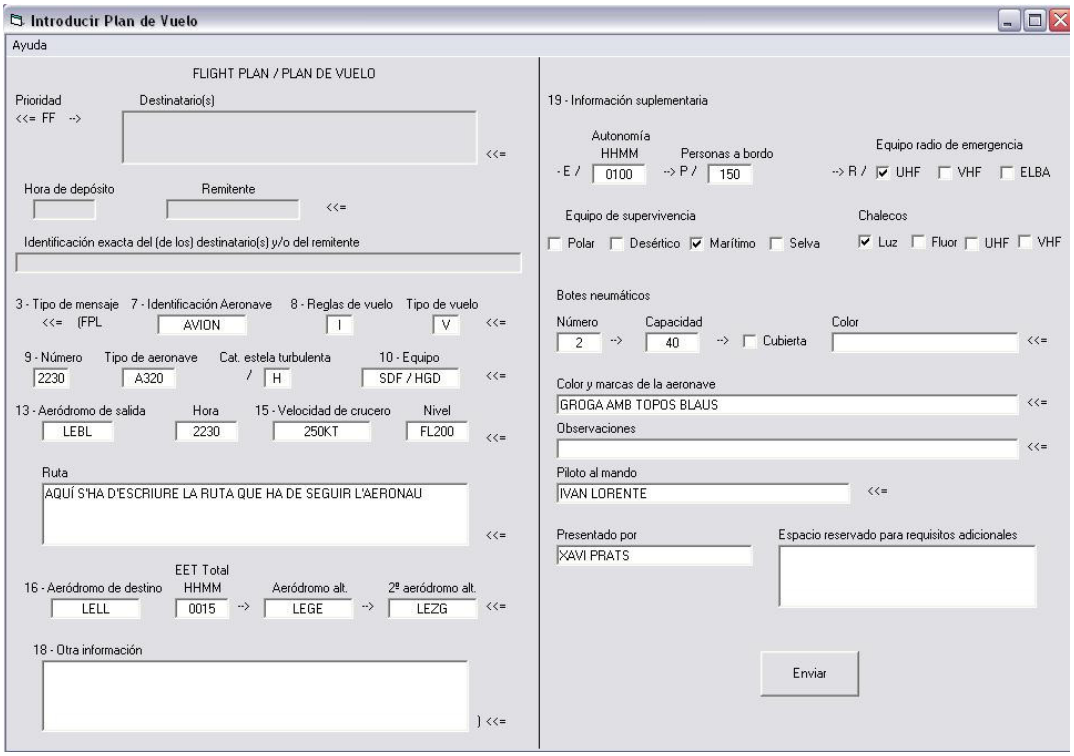


Fig. 3.4 Formulari oficial de Pla de Vol normalitzat

A l'ajuda, podem trobar un arxiu on s'explica acuradament com omplir aquest pla de vol.

En el moment que nosaltres pitgem el botó *Enviar*, l'aplicació guarda totes les dades en una estructura de pla de vol, crea un fitxer i les emmagatzema. A més envia tota l'estructura pel socket cap al servidor. A l'apartat de funcions utilitzades i el protocol de comunicació s'explica com s'envia.

3.3. Estructures de dades

S'ha creat una estructura de dades per a cada tipus de missatge que s'han d'enviar les aplicacions. Tot seguit es comenten els camps que els componen, i es mostra un exemple de les estructures en Visual Basic. Cal dir que al servidor tenen els mateixos camps, però indicant el tamany dels chars (a Visual Basic, el tamany de les cadenes es defineix a la funció que converteix l'estructura en un string per a ser enviat pel socket).

Cal comentar que a tots els camps hi ha com a primer camp definit el nom de l'usuari ja que sempre pot anar bé a l'hora de voler implementar algun altre servei, que necessiti saber el nom d'usuari per alguna raó.

3.3.1. Tvalida

Tvalida és l'estructura definida per, una vegada creada la connexió entre el client i el servidor, donar accés al client als serveis oferts pel servidor.

Les dades que es transmeten són el nom d'usuari i el password introduïts a *Client.frm*, a més d'un últim caràcter que indica al servidor quin tipus d'usuari és (A=ATC, P=Pilot) i indica a l'usuari si ha estat validat (S) o no(N).

Exemple:

```
Public Type Tvalida
   user As String     'Nom d'usuari
   pass As String     'Password del client
   val As String      'Pilot->servidor: A si ATC, o P si Piloto; servidor->client: S
                      si validat o N si denegat
End Type
```

3.3.2. Tmetar

És l'estructura que indica al servidor que volem un metar determinat, el que hi ha guardat al camp metar. Potser no hagués calgut fer una estructura per enviar aquesta dada, però d'aquesta manera totes les dades viatgen de la mateixa forma i mitjançant els mateixos estris, com ara les funcions de conversió d'estructura a string i viceversa.

```
Public Type Tmetar
   user As String
   metar As String
End Type
```

3.3.3. Tnotam

És molt semblant a l'anterior, però amb la particularitat de que quan demanem un notam, ho fem demanant els notams vàlids a una data i una hora

determinats. Es pot donar la casualitat de que hi hagi més d'un notam actiu per un moment determinat.

Al camp *not*, és on hi trobarem la cadena que ens enviarà el servidor amb els notams (1 notam a cada missatge).

```
Public Type Tnotam
   user As String
   airport As String
   fecha As String
   hora As String
   not As String
End Type
```

3.3.4. Tpos

Aquesta estructura correspon a la informació referent a l'aeronau. En ella podem veure camps amb posicions definides pel radar i altres enviats per l'avió. A la pràctica seran iguals els camps x_radar i x, ja que al ser una simulació l'ATC no pot detectar les aeronaus, però s'han definit per si en un futur es vol implementar. Al costat de cada camp, hi ha una definició de què és.

```
Public Type TPos
   'camps simulats pel radar
   user As String #usuari
   x_radar As Long #x observada al radar (en realitat serà la mateixa que la x)
   y_radar As Long
   z_radar As Long
   altitud_radar As Long #altitud observada pel radar

   'camps enviats per l'aeronau
   radio As Long #freqüència de radio utilitzada al moment
   transponder As Long #codi reservat a l'aeronau per facilitar la feina als ATC
   ident As String #identificador de l'aeronau
   velocidad As Long #velocitat en KT observada als comandaments
   x As Long #posició a l'eix x observada a l'aeronau
   y As Long
   z As Long
   altitud As Long
   nwp As String #next way point (pròxim punt de ruta)
   dnwp As Long #distance to nwp (distància al nwp)
   eta As Long #estimated time of arrival to nwp (temps estimat d'arribada al nwp)
   heading As Long #rumb
   track As Long #rumb vertader - derrota
   grspeed As Long #velocitat respecte al terra
End Type
```

3.3.5. Tpv

Aquesta és l'estructura utilitzada per emmagatzemar la informació del pla de vol. Els camps són els mateixos que es poden observar a la figura del formulari *intro_planvuelo.frm*, amb la incorporació de 2 camps més: un d'Estat del pla de

vol (acceptat, espera, o denegat) i un camp de comentaris per a poder explicar en el cas de denegació, les causes i motius. Aquests 2 camps són els utilitzats al formulari *Estado_planvuelo.frm* per mostrar aquesta informació.

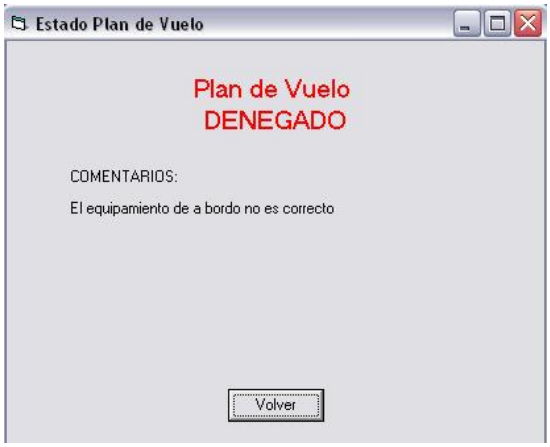


Fig. 3.5 Estat del Pla de Vol

3.4. Descripció de les funcions utilitzades

En aquest apartat es defineixen les funcions que han utilitzat tant servidor com clients pilot per tal de poder comunicar-se.

3.4.1. Servidor

3.4.1.1. int ejecuta_metar()

A l'apartat 3.2.1.1, on es tractava sobre el fitxer server.c i la seva funció main, s'ha explicat que una de les funcions que es cridava era la d'activació d'un procés fill que executes cada X segons.

Aquesta funció crea un procés fill i retorna al main l'identificador d'aquest procés fill.

El procés fill (procés 1) crea cada X temps un altre fill (procés Y) que executa l'aplicació Metar descrita al capítol 5 i mor. El procés 1 torna a esperar a que passi el temps X per tornar a crear un procés Y que farà el mateix que abans.

Queda molt més entenedor en un diagrama de flux.

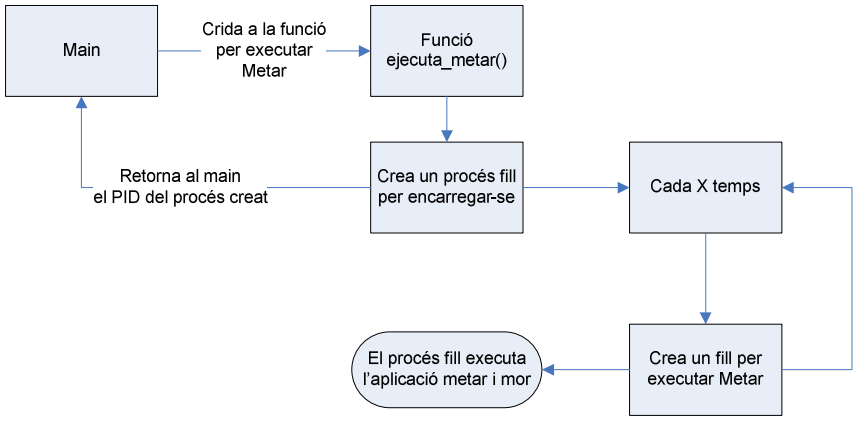


Fig. 3.6 Diagrama de flux de la funció *ejecuta_metar()*

3.4.1.2. int atiende(int sock_listen)

Aquesta funció és cridada pel servidor per tal d'atendre les connexions dels clients. Com en el cas anterior, la funció crea un procés fill que s'encarrega de la gestió dels usuaris, mentre que retorna al servidor el PID d'aquest nou procés.

Aquest procés fill espera a que hi hagi una connexió i queda bloquejat a l'*accept* del socket esperant-la. Una vegada rep una connexió d'un client, crea un procés específic per aquell fill i torna a espera una altra connexió. És la manera en que funcionen els servidors concurrents, i així poden servir i atendre varies connexions a la vegada.

El nou procés, creat específicament per a un client en concret, crea un bucle infinit (si no hi ha cap error ni el client es desconnecta) en el qual serveix totes demandes del client, mitjançant una crida a la funció *servei(sock_conn)*.

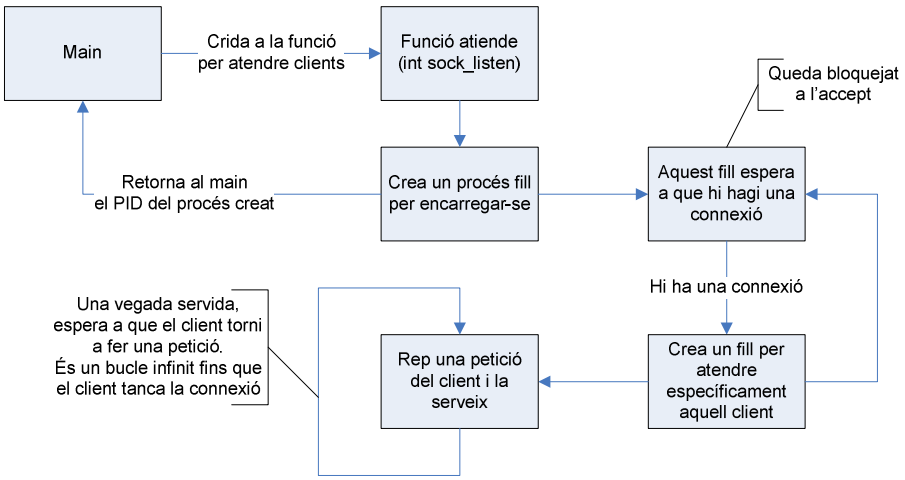


Fig. 3.7 Diagrama de flux de la funció *atiende(sock_listen)*

3.4.1.3. int servei (int sock_conn)

Aquesta funció és l'encarregada d'atendre cada petició d'un client, i enviar-li la informació demandada.

A la funció se li passa el descriptor de la connexió i espera a que li arribi alguna cosa. Quan rep dades per un socket, llegeix el primer byte i determina l'acció a prendre. El protocol que indica quin caràcter està explicat a l'apartat 3.5 d'aquest mateix capítol.

Depenent de quin sigui aquest primer byte, llegirà la resta de la informació emmagatzemant-la en un tipus d'estructura adient. És a dir, si detecta mitjançant aquest primer byte que li ha d'arribar un paquet de tipus Tvalida, guarda les dades directament en una estructura d'aquest tipus i executa l'acció corresponent, que en aquest cas seria, obrir el fitxer de passwords, comparar les dades rebudes amb les del fitxer, i enviar-li un missatge de validació o denegació al client.

Per exemple, la funció servei és l'encarregada de realitzar les operacions del servidor (processos de la dreta de la figura).

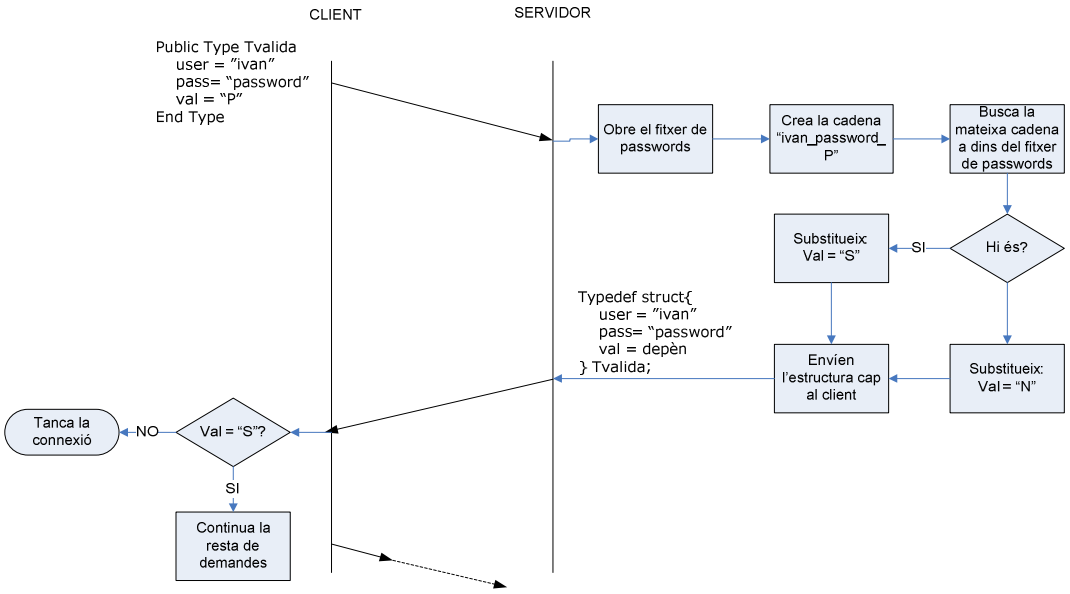


Fig. 3.8 Exemple de funcionament de validació d'un usuari al sistema

3.4.2. Client Pilot

Les funcions implementades al client són més de cara al correcte empaquetament de les estructures en strings, ja que mitjançant el control Winsock de Windows, les funcions SendData i GetData només envien i reben dades emmagatzemades a cadenes de caràcters, i no ho fan directament amb estructures.

Així doncs, s'ha definit una funció per a cadascuna de les estructures de dades, per tal de mapejar correctament aquestes estructures en un string de caràcters per tal de poder enviar les dades i que el servidor les pugui llegir correctament.

3.4.2.1. Enviar dades: Mapejat d'estructures en un string

S'han definit una sèrie de funcions, una per a cada tipus d'estructura, per al mapejat d'una estructura de dades en un string. Aquestes funcions s'han anomenat TstructToStr (tipus d'estructura to string). Per exemple la funció que mapeja l'estructura de posicions s'anomena TPosToStr.

```
Public Function TposToStr(p As TPos) As String
   Dim StAux As String

   StAux = ""
   AgregarString StAux, p.user, 12
   AgregarLong StAux, p.x_radar
   AgregarLong StAux, p.y_radar
   AgregarLong StAux, p.z_radar
   AgregarLong StAux, p.altitud_radar
   AgregarLong StAux, p.radio
   AgregarLong StAux, p.transponder
   AgregarString StAux, p.ident, 6
   AgregarLong StAux, p.velocidad
   AgregarLong StAux, p.x
   AgregarLong StAux, p.y
   AgregarLong StAux, p.z
   AgregarLong StAux, p.altitud
   AgregarString StAux, p.nwp, 15
   AgregarLong StAux, p.dnwp
   AgregarLong StAux, p.eta
   AgregarLong StAux, p.heading
   AgregarLong StAux, p.track
   AgregarLong StAux, p.grspeed

   TposToStr = StAux 'devolvemos la estructura Tpos formateada en un string
End Function
```

Fig. 3.9 Exemple de funció de mapejat d'estructures

A aquesta funció se li passa per paràmetre una estructura de tipus Tpos (p) i retorna un string amb l'estructura mapejada en ell.

Defineix un string anomenat StAux i l'inicialitza buit. A partir d'aquí comença a introduir les dades a l'string, depenent si el tipus de dades a mapejar és string o long. Si el tipus és string, fa una crida a la funció AgregarString, i li passa per paràmetre l'string auxiliar, el camp de l'estructura a mapejar, i el tamany que ha de tenir aquell camp. És molt important el tamany, ja que ha de coincidir perfectament amb els tamanys de l'estructura de dades al servidor, ja que si no, al llegir l'string, el servidor no serà capaç de "demapejar" l'string a l'estructura corresponent. Així doncs, s'ha tingut molta cura a que els tamanys dels strings coincideixin amb els tamanys definits al servidor.

La funció AgregarString és la següent:

```
Public Sub AgregarString(ByRef St As String, Poner As String, Tamaño As Long)
   Dim aux As String

   aux = RTrim(Poner) & Chr(0)
   Do While Len(aux) < Tamaño
      aux = aux & Chr(0)
   Loop
   St = St & aux
End Sub
```

Fig. 3.10 Funció AgregarString

Aquesta funció crea un string auxiliar, i hi guarda la informació que li passem a *Poner*. Com que l'estructura ha de tenir un tamany concret, omplim la resta de caràcter fins a arribar al tamany necessari amb el caràcter 0, i així, a més, tindrem una manera de detectar a C el final d'string.

És a dir, d'aquesta manera, si al servidor hi ha un camp d'una estructura definit com Char user[12] i nosaltres volem enviar-hi "ivan" (4 caràcters), la funció AgregarString guardarà a la cadena auxiliar "ivan\0\0\0\0\0\0\0\0". D'aquesta manera, el servidor al llegir l'estructura guardarà els 12 caràcters al camp correctament, i si a més fem un printf, detectarà que la cadena té 4 caràcters útils.

Respecte a la funció AgregarLong, cal destacar que tan els Long a Visual Basic, com els Integers a C, ocupen 4 bytes. Així doncs, el que fa aquesta funció és guardar aquest tipus de dades en els 4 bytes corresponents.

```
Public Sub AgregarLong(ByRef St As String, ByVal Poner As Long)
   St = St & Chr(Poner Mod 256)
   Poner = Poner \ 256
   St = St & Chr(Poner Mod 256)
   Poner = Poner \ 256
   St = St & Chr(Poner Mod 256)
   Poner = Poner \ 256
   St = St & Chr(Poner Mod 256)
End Sub
```

Fig. 3.11 Mapejat d'un long en un string

3.4.2.2. Rebre dades: Mapejat d'un string en una estructura

De la mateixa manera que a Visual Basic hem definit les funcions per a mapejar una estructura de dades en un string, hem de fer-ho a l'inrevés per tal de, al rebre les dades, poder mapejar-les a l'estructura de dades corresponent.

Tot seguit es presenten les funcions que realitzen aquesta tasca de manera inversa a les explicades a l'apartat anterior.

Cal comentar, però que igualment que al cas anterior, necessitem una funció del tipus StrToTstruct (una per a cada estructura) que cridi a les funcions següents.

```
Public Function LeerString(ByRef St As String, Tamaño As Long) As String
   Dim StAux As String
   StAux = Left(St, Tamaño)
   St = Mid(St, Tamaño + 1)

   Dim Onde As Long
   Onde = InStr(1, StAux, Chr(0))
   If Onde <> 0 Then StAux = Left(StAux, Onde - 1)
   LeerString = StAux
End Function


Public Function LeerChar(ByRef St As String) As String
   LeerChar = Left(St, 1)
   St = Mid(St, 2)
End Function
```

Fig. 3.12 Mapejat d'un string en una determinada estructura de dades.

3.5. Disseny del protocol de comunicacions: transmissió d'informació

Un protocol de comunicacions no és més que una sèrie de normes o regles per tal de que 2 o més màquines siguin capaces de comunicar-se i "entendre's".

La comunicació entre les màquines s'estableix mitjançant la connexió de sockets o canals de comunicació. Els sockets poden connectar-se mitjançant 2 protocols de transport: TCP i UDP.

El primer permet saber del cert que tota la informació arriba al destí i arriba bé, gràcies als seus processos de control, mentre que el segon no ens assegura que arribin totes les dades, però sí que les que arriben són correctes.

Així, s'ens planteja la decisió de quin dels 2 utilitzar. A nosaltres ens interessa molt saber que totes les dades que enviem arribin al destí (tant al servidor com als clients), ja que no poden haver-hi errors a l'enviar un pla de vol, o un metar. Així doncs escollim el protocol TCP com a protocol de transport entre clients i servidors. El protocol UDP es pot utilitzar entre el simulador de vol i l'aplicació client, especialment per a la comunicació de dades com ara les posicions de l'aeronau, ja que han de ser actualitzades cada segon o potser menys temps, i perdre un paquet de dades no és tan preocupant ja que en menys d'un segon arribarà un altre actualitzat.

Una vegada escollit el protocol de transport, ens centrarem al protocol de comunicació entre les màquines. El protocol utilitzat per a comunicar els clients amb el servidor és molt senzill:

Es tracta d'identificar cada paquet amb un determinat byte, per a que les aplicacions sàpiguen quin tipus de paquet és el que està arribant. En el nostre cas s'indica mitjançant una lletra. Quan enviem un missatge pel socket, estem enviant un string o cadena de bytes. Es tracta de que el primer byte d'aquesta cadena, sigui un caràcter que, una vegada llegit per la màquina destí, aquesta pugui discriminar de quin tipus és i pugui tractar el paquet en conseqüència.

En el nostre cas, tant les aplicacions client com l'aplicació servidor saben que si el primer byte del paquet de dades que està arribant és

- "V", és un paquet de validació d'usuari, i per tant, el servidor al rebre'l ha de consultar al fitxer de passwords si l'usuari, el password i el tipus d'usuari (pilot o controlador) coincideixen amb la informació que està arribant tot seguit pel socket. Els clients, al rebre un d'aquests paquets, saben que el que hi ha darrera d'aquest byte és la validació o denegació d'accés al sistema.
- "M", és un paquet de metar. El client sap que a les dades posteriors trobarà el nom de l'aeroport d'on es vol conèixer la meteorologia, i el busca i li torna. El client sap que quan el servidor li envia una "M" a la primer byte de la cadena, darrera trobarà la informació meteorològica que havia demanat.
- "N", és un paquet de notam. El funcionament és el mateix que amb el de metar.
- "X", és un paquet de posicions, que indica al servidor, que la resta de dades del paquet indicaran al servidor la posició de l'aeronau.
- "P", és un paquet de Pla de Vol. El servidor sap que en aquest paquet li està arribant la informació continguda a un Pla de Vol normalitzat que li envia un client. El client sap que quan el rep, trobarà la informació d'acceptació del Pla de Vol o denegació del mateix, i en aquest últim cas les raons de per què no ha estat acceptat.
- "C", és un paquet de chat. El servidor sap que la resta del missatge és una cadena que s'ha de distribuir per la resta de clients, i els clients saben que quan reben aquest tipus de paquet, han de mostrar per pantalla la seva informació.

Un protocol de comunicacions és tan senzill com el vist. Es tracta només d'indicar a les màquines que es volen comunicar quin tipus de paquet i/o d'informació els hi està arribant i com han d'actuar en conseqüència.

3.6. Comunicació amb un simulador de vol: X-Plane

El desenvolupament d'aquesta aplicació client/servidor permet la utilització de diversos simuladors de vol i no està dissenyada per a cap en concret com veiem que passava al software analitzat. Només s'hauria d'incloure un petit

mòdul de comunicació, depenent de cada simulador, per tal de comunicar el simulador de vol amb el client pilot desenvolupat en aquest projecte.

Com s'ha comentat anteriorment, la informació sobre la posició de l'aeronau i la resta de la informació de l'estructura TPos, prové directament del simulador de vol. Aquesta informació es pot extreure o introduir del/al simulador depenent de quin sigui aquest. Pel Microsoft Flight Simulator existeix una aplicació, FSUIPC, que permet realitzar aquesta funció.

A l'inici de la realització del projecte es va fer un petit estudi sobre aquesta capacitat d'extracció i introducció de dades a un simulador de vol anomenat X-Plane.

En aquest apartat es parlarà de com es comunica aquest simulador amb altres màquines.

L'X-Plane, a la seva versió 7.0, permet la introducció i extracció de dades directament des de la seva interfície gràfica. A l'apartat data input/output existeix la capacitat d'indicar quines dades es volen enviar des del simulador i cap a on.

A l'apartat Input/Output comentat es pot veure la distribució de la informació en índexs. Aquests índexs van des de el 0 fins al 107 (en el cas de la versió 7.0 del simulador). I cada índex correspon a diversa informació. La correspondència dels índexs amb la informació que contenen es pot veure a una taula de l'Annex d'aquest projecte.

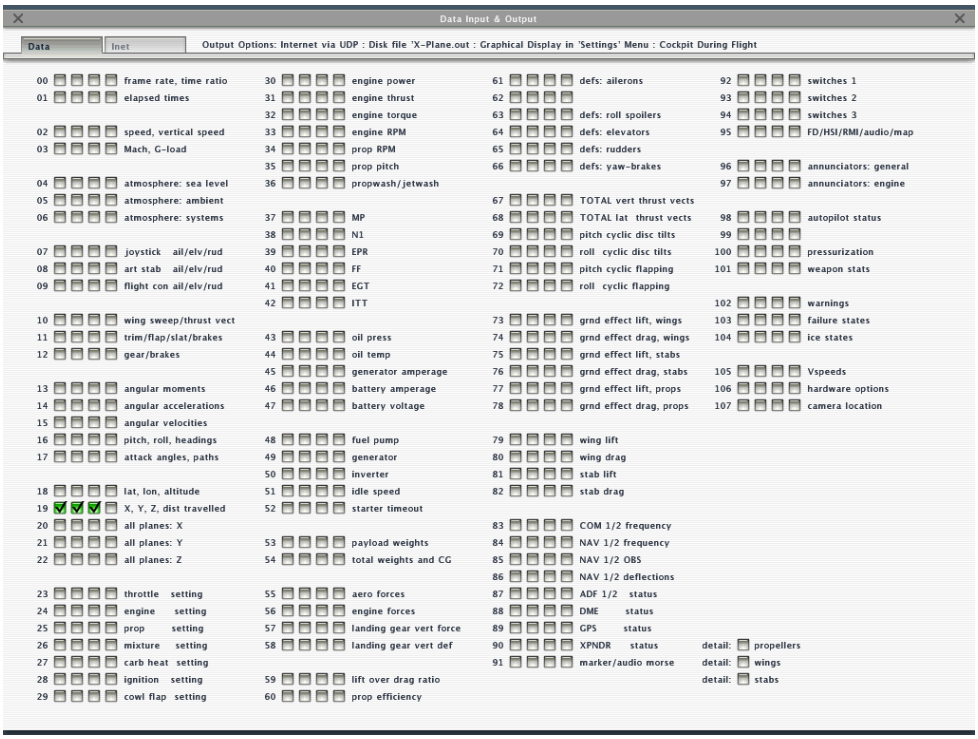


Fig. 3.13 Captura de pantalla Input/Output de l'X-Plane 7.0

X-Plane utilitza el protocol UDP per enviar i rebre dades.

Si amb un analitzador de protocols analitzem un paquet de dades envia't per X-Plane, veurem alguna cosa semblant a això:

68 65 84 65 0 48 33 0 0 0 76 87 67 69 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 48

Aquesta informació s'estructura de la següent manera:

Els 4 primers bytes, 68 65 84 65, corresponen a "DATA", que ens indica que és un paquet de dades que s'envia des de l'X-Plane.

El següent byte és un 0 per defecte.

El 48, ens indica quina plataforma està fent servir X-Plane. El 48 ("0") ens indica que està funcionant a un PC. Si fos un 49 ("1") indicaria un Mac.

Tot seguit arriba un grup de 36 bytes corresponents a la informació pròpiament dita. Els 4 primers corresponen a l'índex esmentat (en format integer), i els altres 32 bytes són la informació corresponent a aquest índex (depenent quina informació sigui seran integers o floats).

Darrera d'un segment de dades pot venir un altre segment de dades o una altra vegada el símbol de la plataforma utilitzada, que ens indica que el final del paquet.

Per introduir dades a l'X-Plane tant sols se li ha d'enviar un paquet igual que aquest però amb l'índex i les dades modificades al nostre gust.

Una bona manera de comunicar l'X-Plane mitjançant UDP i Visual Bàsic és explicada molt bé per Jeff Lewis a la seva pàgina web (veure [17]).

4. INFORMACIÓ METEOROLÒGICA AERONÀUTICA *METAR*

4.1. Definició de metar

Un METAR és un informe meteorològic aeronàutic codificat que ens proporciona la meteorologia regnant a un aeroport determinat en un moment donat. Les dades d'aquest metar són obtingudes de la estació meteorològica local ubicada a cada aeroport. Aquestes observacions es realitzen a intervals horaris fixos (a Catalunya i a Espanya normalment es difonen cada hora o cada mitja hora, segons els Institut Nacional de Meteorologia) o, fora de programa, si la situació meteorològica ho requereix degut a baixa visibilitat, precipitacions o tempestes (a aquests últims s'els anomena SPECI en comptes de metar). El TAFOR és un informe similar al metar, però amb la diferència que aquest ens informa de la previsió meteorològica per a les següents hores.

Els metars són demandats per les aeronaus, o els diferents usuaris aeronàutics, als Serveis de Trànsit Aeri, abans i durant un vol, per tal de poder organitzar i planificar la sortida i l'arribada a un determinat aeroport. L'Institut Nacional de Meteorologia també presta un servei d'informació meteorològica codificada oficial (metar) a través d'Internet a través del AMA (Autoservei Meteorològics Aeronàutic http://ama.inm.es)

4.2. Estructura d'un metar

Un metar és un informe meteorològic codificat, que pot ser molt senzill o extremadament complicat. Tot seguit utilitzarem una estructura genèrica d'un d'aquests informes, on s'ha separat cadascuna de les seves parts mitjançant el codi cromàtic següent. Després s'aplicarà aquests subdivisió del metar en un exemple real d'un informe de l'aeroport de Barcelona.

L'estructura d'un metar es pot dividir en 7 parts ben diferenciades que facilitaran la comprensió dels seus significats. Utilitzarem codis cromàtics per diferenciar bé aquestes 7 parts i fer la seva comprensió encara més fàcil.

Codi de l'aeròdrom	Temperatura i punt de rosada
Hora	QNH: Ajust de l'altímetre
Vent	Canvis esperats
Visibilitat	

Fig. 4.1 Codi cromàtic per comprendre l'estructura d'un metar

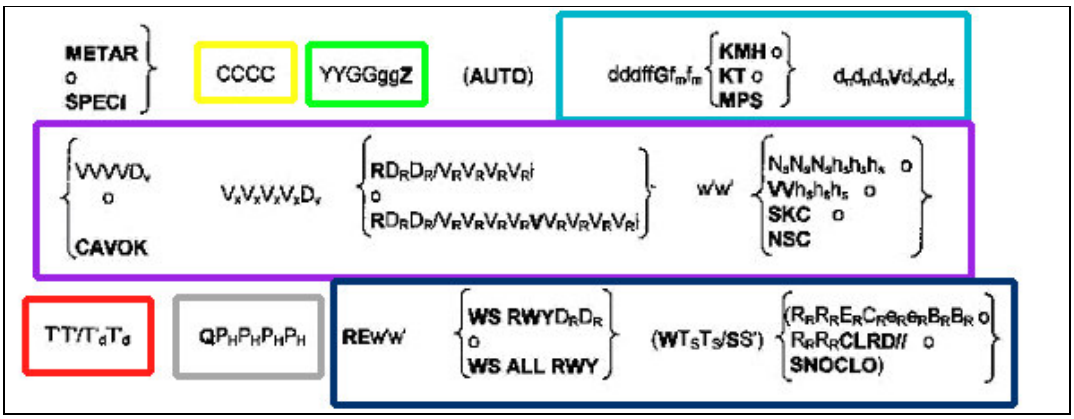


Fig. 4.2 Exemple genèric d'un metar diferenciant les seves diferents parts

4.3. Significat de cada camp

Vegem ara el significat exacte de cada camp. Les unitats dels informes poden variar a diferents regions del planeta. Així tenim que la visibilitat a EEUU s'estableix amb SM ("Statute Milles") mentre que a Europa ho fem en metres. De totes maneres nosaltres ens centrarem amb l'estudi amb unitats i METARS europeus.

4.3.1. Aeròdrom

És el codi de l'aeròdrom que crea el METAR en format OACI (4 lletres)

4.3.2. Hora

L'hora Zulú és el Temps Internacional UTC (Universal Coordinated Time) o GMT (Greenwich Meridian Time). Per obtenir l'hora espanyola li hem de sumar dues hores a l'estiu i una a l'hivern.

Les hores de Sortida i d'Arribada, les autoritzacions dels ATC (Control de Trànsit Aeri), les hores dels informes METAR i altra informació en aviació, on l'horari és clan i fonamental, s'expressen en horari universal UTC.

En un metar, els dos primers dígits de la hora, corresponen al dia del mes actual; i els 4 restants a la hora UTC en que es va emetre aquell informe meteorològic. La Z ens indica que els temps donat és UTC

4.3.3. Vent

Aquest tercer apartat ens informa del vent que hi ha a l'aeròdrom i, si n'hi ha, quines característiques té.

Els 3 primers dígits fan referència a la direcció vertadera des d'on bufa (arrodonit a 10 graus o **VRB** si la direcció del vent és variable).

Els dos últims dígits fan referència a la velocitat en nusos (mph) que són indicats amb les sigles KT ("Knot") al cas d'Europa.

Si existeixen ràfegues de vent ("Gust" en anglès) s'afegeix una **G** seguida de 2 dígits indicant la velocitat d'aquestes ràfegues.

Si el vent canvia sobtadament la direcció o la velocitat s'afegeixen les sigles **WS** ("Wind Shear"). Aquest cas, però, és molt estrany.

Si el vent canvia la seva direcció en 60 graus o més s'afegeix una **V** de Variabilitat.

Com a cas particular direm que si hi ha calma total el vent s'expressa com a 00000KT

4.3.4. Núvols i visibilitat

La visibilitat és un paràmetre molt variable (molt més que el vent) i hi ha vegades que la meteorologia té unes determinades característiques i altres vegades té unes completament diferents. Així doncs, cal dir, que els fenòmens meteorològics que no es donen no tenen presència al metar, lògic per una altra banda.

En el cas de visibilitat total el camp de visibilitat és omplert amb **CAVOK** ("Ceiling and Visibility OK"), que vol dir que tenim el cel sense núvols i la visibilitat horitzontal és més de 10000 metres.
Excepte en el cas just mencionat, la visibilitat horitzontal és obligatòria. Aquesta, a Europa, s'indica en metres i va des de els 1000 fins als 9999. Als EEUU per exemple s'utilitzen les milles (5SM, per exemple)

A partir d'ara s'inicien una sèrie de camps que poden o no estar a l'informe, i van amb el següent ordre:

- Abast visual a la pista: és la distància horitzontal des que el pilot ha de poder veure la pista quan s'aproxima a ella. Aquest camp s'indica amb una R, el número de pista, quina pista (L –esquerra–, R –dreta– , o C –central–), un "/" i la distància a la pista.

  Exemple: R28L/1200 vol dir que l'abast visual a la pista 28 esquerra és de 1200 metres.

- El cel sense núvols, o "SKy Clear", s'indica amb les sigles SKC. Sol indicar-se quan per alguna raó la visibilitat no pot ser CAVOK, però la distància horitzontal és normalment, més gran de 8000 metres.

- Precipitacions: Són les possibles precipitacions que hi hagi a l'aeròdrom. Les múltiples possibilitats de precipitació són descrites a la taula de més avall. Evidentment per a que hi hagi precipitacions hi ha d'haver-hi núvols.

- Enfosquiment: Pot ser que el cel s'enfosqueixi sense la necessitat de que hi hagi núvols (com per exemple que hi hagi fum o pols), hi això s'ha de poder representar. A la taula de més avall hi ha una relació de possibles casos d'enfosquiment.

- Núvols: Es representen de 4 formes diferents:

    - FEWXXX si hi ha núvols escassos (cobertura del cel entre 1/8 i 2/8) a l'altitud indicada per XXX.

    - SCTXXX si hi ha núvols dispersos (cobertura del cel entre 2/8 i 4/8) a l'altitud indicada per XXX.

    - BKNXXX si hi ha nuvolositat abundant (cobertura del cel entre 5/8 i 7/8) a l'altitud indicada per XXX.

    - OVCXXX si tenim el cel totalment cobert a l'altitud indicada per XXX.

  Podem considerar el cel com a capes, així que a diferents altituds podem tenir diferents tipus de nuvolositat. És a dir, que tots 4 tipus de núvols poden conviure en un mateix metar.

CALIFICACIÓ:	DESCRIPTOR:	PRECIPITACIONS:	ENFOSQUIMENT:
- Suau	MI Baixa	DZ Pluja fina	BR Boirina
+ Forta	BC Bancs	RA Pluja	DU Pols difusa
(ejs: - RA Pluja suau	DR Ventisca	SN Neu	DS Vendaval de pols
+ DR Forta ventisca)	FZ Congelació	IC Gel	FC Tromba
	PR Parcialment	GR Calamarsa	FG Boira
	SH Ruixats	GS Pedregada	FU Fum
	TS Tempesta	SG Neu en flocs	HZ Calima o bruma
		UP Precipitació desconeguda	SA Sorra
			SS Tempesta de sorra
			VA Cendres volcàniques

Fig. 4.3 Tipus de precipitacions i enfosquiments i intensitats dels mateixos

4.3.5. Temperatura i punt de rosada

Els dos primers dígits ens indiquen la temperatura de l'aire en graus Celsius i els dos últims el punt de rosada (temperatura a la qual s'ha de refredar l'aire per a arribar a la saturació -100% d'humitat relativa-).

Quant més s'aproximi la temperatura del punt de rosada a la temperatura de l'aire existeix més possibilitat de la formació de núvols, boires i precipitacions.

A Espanya el punt de rosada sol variar entre els 0 i 10 graus per sota de la temperatura de l'aire. És evident que a zones més humides el punt de rosada s'aproxima molt al de la temperatura de l'aire.

4.3.6. QNH

Indica la pressió atmosfèrica a la que s'ha de calibrar l'altímetre de les aeronaus per a que reflexi l'altura real de l'aeroport sobre el nivell del mar (QNH).

Pot estar expressada en HectoPascals a Europa (per exemple Q1020) o "inches" a EEUU (per exemple A2900)

4.3.7. Canvis esperats

Aquesta darrera part d'un informe de meteorologia aeronàutica esdevé una petita previsió en quant a les característiques atmosfèriques en les pròximes hores.

Si no es preveu cap canvi en les pròximes hores, aquest camp val NOSIG (canvis no significatius).

Si es preveu algun canvi, aquesta part pot tenir els següents valors:

- BECMG XXYY: Es preveu un canvi entre les XX i les YY
- TEMPO XXYY: Fluctuacions entre les XX i les YY
- PROBZZ XXYY: Probabilitat del ZZ% entre les XX i les YY
- FM XX: Inici d'un canvi significatiu a partir de les XX
- RMK: Comentari
- AO: Observació automatitzada

4.4. Exemple d'un METAR

En un exemple senzill és molt fàcil de veure el significat:

LEBL 282204Z 32005G15KT 3000 FG FEW002 SCT025 10/10 Q1030 NOSIG

LEBL Ens indica que el metar és de l'aeroport de Barcelona (LEBL)

282204Z S'ha creat el dia 28, a les 22:04 hora Zulú (23:04 hora de Barcelona)

32005KT La direcció del vent és de 320º, amb una força de 5KT però, existeixen ratxes de fins a 15KT

3000 FG FEW002 SCT025 La visibilitat horitzontal és de 3000 metres. Hi ha boira a l'aeroport. Hi ha núvols escassos a 200 peus d'altitud i núvols dispersos a 2500 peus.

10/10 La temperatura de l'aire és de 10 ºC i el punt de rosada també (100% d'humitat relativa)

Q1030 L'ajust de l'altímetre s'ha de situar a 1030 HectoPascals

NOSIG No s'esperen canvis significatius en les pròximes hores

5. DISSENY DE L'APLICACIÓ *METAR*

5.1. Descripció de l'aplicació

5.1.1. Objectiu de l'aplicació

L'objectiu principal de l'aplicació generadora d'informació meteorològica aeronàutica o *metar*, és precisament la de crear aquests informes meteorològics per tal de poder utilitzar-los a la xarxa virtual de simulació de vol, cada vegada que és executat.

Aquesta aplicació és capaç de generar automàticament i gràcies a uns paràmetres donats, els informes *metar* a l'àmbit espanyol (o de qualsevol altre país amb característiques meteorològiques semblants a les nostres, com ara bé Itàlia o Grècia).

Aquesta aplicació s'utilitza conjuntament amb el servidor creat a la primera part d'aquest projecte. El servidor crea cada 30 minuts un procés fill que crida a l'aplicació per a que generi un metar nou. Evidentment, modificant el codi font del procés llençat pel servidor per a crear els metars es pot variar la freqüència de creació d'aquests informes, encara que no és recomanable.

La principal virtut d'aquesta aplicació és que una vegada hi ha un metar generat, el pròxim es crearà en base a ell, i no farà un altre metar aleatori si l'últim creat té una antiguitat de menys d'una hora. Això vol dir que una vegada s'ha creat un metar aleatòriament, el següent (per defecte al cap de 30 minuts) metar que crearà no serà aleatori, si no que agafarà les característiques de l'anterior i les modificarà per tal de que no hi hagi canvis meteorològics molt importants en un breu espai temporal. Si l'aplicació detecta que l'últim metar està creat fa més d'una hora (per exemple que s'hagi acabat la classe de pràctiques, s'hagi aturat l'aplicació i s'hagi tornat a engegar el dia següent) sí que crearà un metar nou aleatori, ja que es considera que l'últim no respon a les característiques meteorològiques de fa poc temps.

Per aconseguir aquest objectiu, per a cada aeroport, l'aplicació crea varis fitxers:

1. Un de text diari amb el següent format: XXXXAAMMDD.txt

   Aquests fitxers contenen els metars de l'aeroport XXXX amb data AAMMDD, i són fàcilment consultables al ser un arxiu .txt.

   Aquest fitxer està organitzat amb un encapçalament que indica la data, l'hora i l'aeroport del metar, i una línia més avall el metar pròpiament dit, tal i com es pot veure a la següent figura. Podríem dir que és l'informe amb format "oficial".

```
2005/02/15 14:30 LEBL

LEBL 151430Z 35011KT 9999 FEW018 13/9 Q1004 NOSIG
```

Fig. 5.1 Exemple d'un informe metar "oficial"

2. Un fitxer ocult amb el format: .XXXX (XXXX és el nom de l'aeroport)

Aquest fitxer ocult té un format diferent, capaç de ser llegit pel programa per a poder separar els camps, analitzar-los, i modificar-los. Tal com s'ha esmentat aquest fitxer és només per a l'aplicació. No es crea per als usuaris. El format intern d'aquest fitxer és el següent:

```
#LEBL.151430Z.35011KT.9999 FEW018.13/9.Q1004.NOSIG
#LEBL.151500Z.32014KT.CAVOK.12/9.Q1006.NOSIG
```

Fig. 5.2 Exemple d'un informe metar de "configuració"

Com es pot veure hi ha diverses diferències respecte al fitxer "oficial":

- En primer lloc aquest no porta capçalera.
- No hi ha línies en blanc entre els metars.
- A l'inici de cada línia hi ha el caràcter #
- Entre cada camp hi ha un punt en comptes d'un espai.

Aquestes característiques són degudes separar els camps d'una manera concreta i indicar al sistema on comença cada part del metar, per tal de poder aïllar-les i tractar-les d'una manera independent i sense errors.

5.1.2. Entorn de treball

Aquesta aplicació s'ha desenvolupat en un entorn UNIX, concretament la distribució Suse Linux 9.1, ja que el servidor on anirà integrada aquesta aplicació s'ha programat en el mateix entorn. S'ha utilitzat el llenguatge C de UNIX per a desenvolupar-la.

5.1.3. Diagrama de flux de l'aplicació METAR

Seguidament es mostra un diagrama de flux general del funcionament de l'aplicació. Als annexos es poden trobar els diagrames de flux complets sobre l'aplicació i cadascuna de les seves funcions.

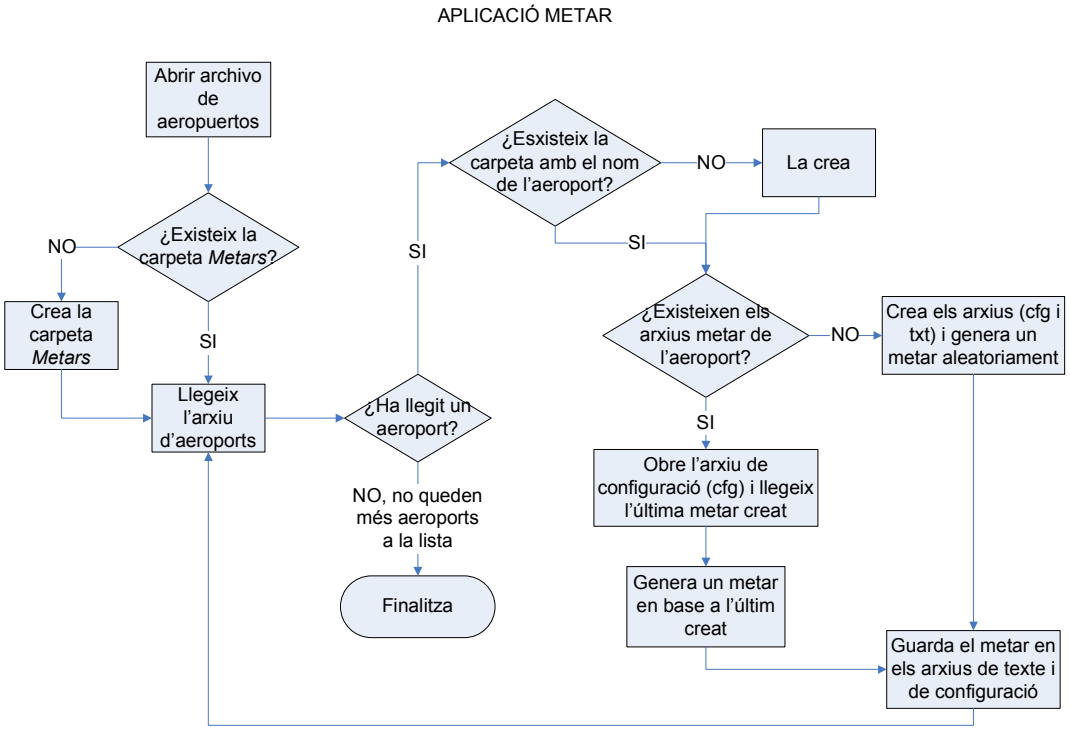


Fig. 5.3 Diagrama de flux de l'aplicació *METAR*

5.1.4. Adaptació dels camps d'un metar

S'ha decidit adaptar algun camp dels informes per tal de facilitar la seva creació ja que en un entorn no professional com ara una classe de pràctiques, no era necessària la inclusió de totes les opcions reals d'un d'aquests informes.

Una de les adaptacions que s'han fet a l'hora de crear un metar és la de no modificar mai el camp de canvis esperats. Així sempre serà NOSIG, indicant que no hi ha canvis significatius. Aquesta decisió és deguda a que el funcionament del programa impedeix grans canvis sobtats en la meteorologia.

S'ha decidit també eliminar algunes de les possibles opcions de precipitacions i enfosquiments. La nostra aplicació funciona amb les següents opcions:

Precipitacions: RA, +RA, -RA, -SN, DZ

Enfosquiments: BR, FG

Aquesta decisió es pren ja que els altres fenòmens atmosfèrics descrits a la taula de l'apartat 6.3.4 són rarament vistos a terres espanyoles, i si els haguéssim inclòs hauríem d'haver-li donat una probabilitat pràcticament nul·la.

5.2. Descripció de fitxers i funcions

En aquest apartat veurem els diferents fitxers que s'han utilitzat per al desenvolupament de l'aplicació que genera informació meteorològica aeronàutica.

5.2.1. Fitxer principal: *Metar.c*

Aquest és l'arxiu principal de l'aplicació i on està el main. La funció main d'aquest fitxer és la següent:

1. Obrir el fitxer d'aeroports (*aeropuertosesp*).
2. Si no existeix, crea una carpeta de nom *Metars*, on es guardaran tots els metars.
3. Llegeix del fitxer d'aeroports un d'ells, i si dintre de la carpeta *Metars* no existeix una altra amb el nom de l'aeroport la crea.
4. Una vegada es situa a dins de la carpeta d'un aeroport, crida la funció *crea_archivos*, que hi ha en aquest mateix fitxer, per tal de crear els arxius metars.

La funció *crea_archivos* és la que s'encarrega de cridar a les funcions de generació dels metars.

Les passes que fa aquesta funció són les següents:

Comprova que existeixin els arxius txt i ocult (format llegible per al programa).

Si no hi són, els crea i crida a la funció *crea_metar* que veurem més endavant per a crear metars nous aleatoris.

Si hi són, obre l'ocult i fa una comprovació entre la data del metar i la data actual.

Si la diferència de dates és major d'una hora, crida a la funció *crea_metar* per a crear un metar nou aleatori, ja que l'últim metar creat és massa antic per a crear un altre en base a ell.

Si la diferència de dates és menor, crida a la funció *analiza_metar* i li passa per referència l'últim metar creat, per a què el modifiqui suaument i no hi hagi canvis molt grans de metar en metar.

5.2.2. Fitxers secundaris

Els dos següents fitxers que veurem són igualment importants per al funcionament correcte de l'aplicació. Realment són l'ànima del programa. El primer, *crea_metar.c*, genera un metar aleatori, mentre que el segon,

analiza_metar.c, observa l'últim metar creat i en base a ells, si considera que és vàlid, construeix el següent.

5.2.2.1. crea_metar.c

Aquesta funció crea un metar d'un aeroport d'una manera pseudo-aleatòria entre uns rangs donats. En aquesta funció s'utilitzen d'una manera molt fluïda les funcions que retornen números aleatoris entre un rang donat amb certa probabilitat que veurem en apartats posteriors.

La funció crea camp a camp totes les parts d'un metar i les ajunta al final per retorna-les a la funció *crea_archivos*.

No esmentarem cada regla per a crear cada part, però la majoria de les condicions i/o fenòmens meteorològics estan creats en base a la climatologia del país.

Els rangs i valors donats a cada paràmetre estan compresos dins dels valors probables corresponents a la climatologia real.

Molts valors com per exemple el QNH es defineixen en base a l'observació de metars oficials de l'Institut Nacional de Meteorologia. S'ha de dir, però, que no reflecteixen cap dada oficial, i que com la meteorologia no és una ciència exacta pot ser que a vegades els valors reals no coincideixin els rangs definits a l'aplicació.

5.2.2.2. analiza_metar.c

Tal i com s'ha explica't a l'apartat 7.2.1 utilitzem 2 fitxers per a crear els metars. La funció *analiza_metar* llegeix els camps del metar des de l'arxiu de metars ocult, ja que és des d'on pot llegir-les correctament.

Una vegada tenim tots els camps per separat, la funció els retoca o els deixa igual com eren.

Com en el cas anterior, no esmentaré cada actuació de la funció, però comentaré aspectes rellevants per tal de mostrar l'eficiència de l'aplicació:

Per exemple, el cas de la visibilitat. És evident, que si la visibilitat horitzontal és baixa és degut a alguna causa, ja siguin més núvols, precipitacions, enfosquiments, etc. Això queda reflectit a l'aplicació. Si per exemple, la visibilitat horitzontal és menor de 5000 metres, és més probable que hi hagi núvols i/o precipitacions que si la visibilitat és major.

Un altre exemple, la temperatura. És evident que als mesos d'hivern la temperatura és més baixa que a la resta d'estacions. I que a la nit les temperatures tendeixen a baixar. Això també queda reflectit al programa.

5.2.2.3. fecha.c

El fitxer fecha.c conté una funció -*funcion_fecha(char* fecha, int opcion)*- que guarda a l'string *fecha* una data amb un format determinat. Les diferents opcions són les següents:

"AAMMDD.txt": Aquest format és necessari per a poder crear els arxius diaris de cada metar.

"DDHHMMZ": Aquest format és el que conté el metar

"DD/MM/AA HH:MM": Aquest format és utilitzat a la capçalera del metar

"MM": Aquest format l'utilitzarem per a saber a quin més som actualment i poder definir correctament les temperatures

"HHMM": Semblant a l'anterior, ens retorna l'hora exacta actual per tal de poder definir correctament les temperatures.

5.2.3. Fitxers i funcions de rutina i ajuda

A continuació es detallen les funcions de rutines cridades per les diferents funcions de l'aplicació metar i les ajudes per a la compilació.

5.2.3.1. rutines.c

En aquest fitxer hi ha compreses les funcions més generals i que poden ser cridades per qualsevol dels altres arxius.

Entre aquestes funcions, que inclouen una funció de gestió d'errors, una d'espera que dorm el procés per un temps determinat, una altra que retorna la ruta o el directori de treball actual, cal destacar una sèrie de funcions per tal d'aconseguir números aleatoris.

5.2.3.2. Funció int randomAB(int A, int B)

Aquesta funció retorna un numero aleatori entre un rang que hem passat per paràmetre. Utilitza la funció de C *rand* per retornar aquest valor aleatori.

5.2.3.3. Funció int randomABprobXXXX(int A, int B, int C)

Hi han definides 3 funcions amb aquest encapçalament, on només canvien les XXXX per: 5060, 6070 i 8090.

Aquestes funcions retornen un número pseudo-aleatori entre dos rangs donats, però amb una certa probabilitat (XXXX) de que el valor retornat sigui proper a

un d'ells. Indicarem a quin d'ells ha de ser proper mitjançant la tercera variable que li passem per referència, que ha de ser un dels altres dos donats.

Aquestes funcions no tenen una base matemàtica constatada però no és necessari ja que l'objectiu és crear metars aleatoris o pseudo-aleatoris, i la meteorologia no és una ciència exacta.

La probabilitat de que el número retornat sigui proper (o més proper a un que a l'altra) s'aconsegueix de la següent manera:

Per aconseguir una probabilitat del 60 o 70% aprox. cap a un dels dos rangs es crida a la funció randomAB que ens retorna un número aleatori entre els dos rangs. Trobem el número mig entre els dos rangs, i el comparem amb el número aleatori retornat per randomAB. Depenent a quin dels dos rangs s'havia d'acostar el número aleatori final, tornem a cridar a la funció randomAB o retornem el número aleatori que ja teníem. Aquest procediment es pot observar al diagrama de flux següent.

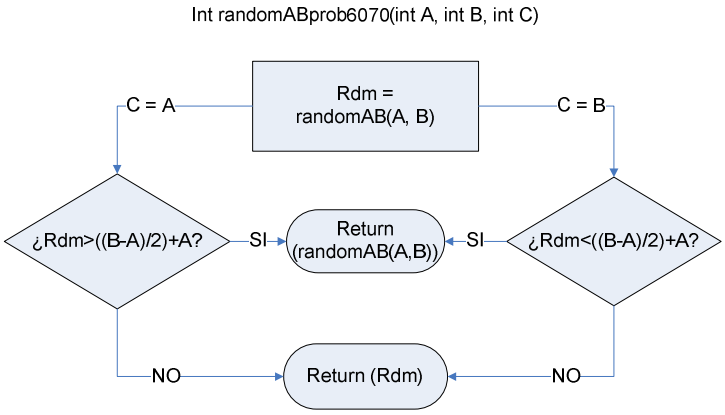


Fig. 5.4 Diagrama de flux de la funció randomABprob6070(int A, int B, int C)

En el cas de les altres dues funcions (5060% i 8090%) el procés és similar però amb algunes petites variacions. Per exemple, en la funció del 50 o 60%, quan el flux segueix el SI, en comptes d'anar a on va al diagrama, va cap a un altre randomAB(0,1) que determina si ha d'entrar al random del mig (0) o no (1).

El procés amb la del 8090% és similar a aquestos dos, però amb alguna petita variació que fa que la probabilitat sigui la esmentada.

Com que no necessitem una base matemàtica, aquestes funcions han estat denominades amb aquestes probabilitats en base a les probabilitats que sortien a les proves realitzades.

5.2.3.4. Makefile

El fitxer *Makefile* és un fitxer per a poder compilar l'aplicació d'una manera senzilla i ràpida. S'utilitza com qualsevol altre fitxer Makefile.

5.2.4. Fitxer aeropuertoesp

Aquest fitxer és necessari per tal de poder crear els metars, ja que és en ell on hi ha la informació dels aeròdroms que volem crear un metar.

En ell trobem una llista de tots els aeroports espanyols en format OACI. El format del fitxer és un codi d'aeroport per a cada línia de text.

Recordem que l'aplicació *metar* està dissenyada per a aeroports espanyols, ja que les característiques d'aquests metars estan definits i dissenyats amb les característiques aproximades de la meteorologia espanyola.

Encara així podríem utilitzar aquesta aplicació en països amb meteorologia similar a Espanya, com ara Itàlia o Grècia, però no podríem utilitzar-lo en països no tan llunyans, com ara França, on les temperatures, per exemple, solen ser bastant menors al nord del país.

Per a poder incloure aeroports d'altres països amb meteorologia similar a l'espanyola, només cal afegir-los a partir de l'últim.

6. PROVES DE FUNCIONAMENT

La implementació de les aplicacions client/servidor i el seu protocol de comunicacions i les proves del sistema s'han anat realitzant de manera conjunta, per tal d'analitzar a la mateixa vegada quins eren els possibles errors i intentar reparar-los el més ràpid possible. Per tal de fer més eficient la implementació del sistema, s'han separat les aplicacions ens diferents parts i cadascuna d'elles s'anava provant per tal d'assegurar el correcte funcionament de cadascuna d'elles. Després d'analitzar el comportament de cadascuna de les parts, s'ha comprovat el bon funcionament global de tota l'aplicació.

Tot seguit s'esmenten les diferents parts que s'han anat provant.

6.1. Arquitectura client/servidor

6.1.1. Servidor

Per a posar en funcionament el servidor es va implementar poc a poc la seva estructura i les seves principals característiques:

S'ha provat quins arguments se li han de passar al servidor a l'hora d'invocar-lo, la bona implementació de les "signals", les funcions que imprimeixen informació com ara el missatge d'execució del servidor, o el missatge de finalització, o el menú d'opcions. En aquest últim cas es va provar el funcionament correcte de les opcions, una per una.

També és va provar el bon funcionament de les funcions que crida el servidor, i les funcions d'ajuda i/o rutines, com ara les rutines d'errors, o d'espera. Es van crear error intencionadament per tal de veure el funcionament correcte.

6.1.1.1. Creació de processos. Servidor Concurrent

S'ha analitzat acuradament el bon funcionament dels processos fills creats pel servidor. Incloem els processos "autònoms" com ara el procés que activa l'execució de l'aplicació Metar cada X segons.

I processos depenent d'altres com els fills creats per a cada client quan arriba una connexió. Per a tal efecte, s'ha provat de connectar diferents clients a la vegada i visualitzar el número de processos fills creats mitjançant la comanda ps de UNIX. D'aquesta manera s'assegura el funcionament concurrent del servidor. Primer amb diverses màquines amb Linux, i posteriorment, i una vegada realitzades les proves de sockets que veurem més endavant, proves amb els clients destinats a tal efecte, creats en entorn Windows amb Visual Basic.

6.1.2. Client

S'ha analitzat acuradament les accions i funcionament del servidor. S'ha provat la correcta admissió de dades per part de l'usuari i el seu emmagatzemat a les diferents estructures de dades, la seva interfície gràfica, el correcte funcionament de les seves funcions (*agregarstring*, *agregarlong*, *eslng*, etc.) per mapejar estructures en un string per a enviar-lo pel socket, etc. En aquest últim cas, s'omplien els camps de les estructures amb dades que podien ser reals, i es provava el seu mapejat en una cadena de caràcters, que és el tipus d'informació que es pot enviar directament per un socket amb el control Winsock.

6.1.3. Sockets

Els sockets s'han anat provant a mesura que s'estaven implementant per tal de provar la correcta connexió entre les màquines. Al principi, i per tal d'ajudar a la implementació d'aquest apartat al servidor, es van provar sockets TCP entre màquines UNIX, per tal de provar si connectaven correctament. Tot seguit, i una vegada implementats els sockets al client en Visual Basic, es van fer proves de comunicació entre sistemes operatius diferents.

6.1.4. Protocol de comunicacions

Durant el disseny i la implementació del protocol de comunicacions es van fer la majoria de les proves. És evident que tenia una importància màxima ja que és la base d'aquest projecte. Es van fer moltes proves degut als problemes existents a l'implementar aquesta aplicació en 2 sistemes operatius diferents i llenguatges de programació diferents. Es van tenir problemes amb la interpretació de les dades, ja que, per exemple, el compilador de C utilitzat per a la creació del servidor, el gcc, no definia les estructures tal i com se l'indicava. Encara que es defineixi una estructura amb diferents camps de diferent tamany, hi ha compiladors que redefineixen el tamany d'aquests camps, i omplen la resta de bytes amb basura sense sentit, i que ens donen molts problemes a l'hora de, per exemple, mapejar l'string arribat per un socket en una determinada estructura i que tots els camps de l'estructura s'omplin amb la informació corresponent, tal i com s'ha enviat des de l'altra màquina.

6.2. Aplicació Metar

L'aplicació generadora d'informes meteorològics aeronàutics ha acaparat moltes de les proves de funcionament de les aplicacions. Aquesta aplicació ha necessitat de moltíssimes proves per a assegurar un bon funcionament.

S'han fet moltes proves de funcionament de les funcions de generació de números aleatoris amb una certa probabilitat.

Així com a la creació de les carpetes i fitxers on es guarden els metars, i la navegació per dins d'aquestes carpetes a partir del directori de treball actual.

També s'han realitzat moltes proves amb les funcions *crea_metar* i *analiza_metar*, encarregades de la creació d'un metar nou, i l'anàlisi de l'últim metar existent i la seva modificació. Per a tal efecte, i com que els metars es poden subdividir en 7 parts diferents, s'ha analitzat el funcionament de les funcions a cada una d'aquestes parts, observant els metars que es creaven, les opcions que es podien afegir o les possibles millores per tal d'acurar el més possible el resultat obtingut amb els possibles resultats a la vida real.

7. CONCLUSIONS

Una vegada finalitzada la implementació del sistema i les diferents aplicacions, s'avaluaran els resultats obtinguts i s'extrauran conclusions.

En aquest capítol es valora fins a quin punt s'han complert les expectatives inicials, indicant les possibles causes de desviació. Es comprovaran si s'han obtingut els resultats esperats realitzant un balanç dels objectius fixats en un principi i es plantegen possibles millores i ampliacions futures. Finalment s'exposen les conclusions personals sobre la realització d'aquest Projecte.

7.1. Objectius assolits

Pràcticament s'han assolit totes les tasques i funcionalitats definides als apartats d'objectius de cadascuna de les aplicacions, tot i que han quedat algunes tasques per finalitzar com ara la memòria compartida per tal de comunicar els processos fills entre ells, per la implementació, per exemple, d'un xat entre els diferents clients, o la possibilitat de poder visualitzar quants i quins clients estan connectats a la vegada al servidor (recordem que el procés que gestiona el menú del servidor és diferent al procés que gestiona les connexions, i per tant els processos no es poden comunicar sense una memòria compartida).

7.2. Desviació sobre el projecte inicial

A l'hora d'escollir el projecte, s'havia fixa't com a objectiu l'estudi de viabilitat d'un protocol de comunicació entre simuladors de vol, i es va escollir un determinat simulador de vol per tal de realitzar totes les proves. Després però, es va veure que seria més útil si aquesta xarxa de simulació de vol es feia de forma genèrica i que pogués funcionar amb diferents simuladors de vol, implementant només un petit mòdul de comunicació a l'aplicació client, individual per a cada simulador de vol, per tal d'extreure o introduir-ne dades del mateix.

7.3. Millores possibles i ampliacions futures

La principal millora és la finalització de la implementació de la memòria compartida al servidor per a poder implementar altres serveis, com per exemple l'esmentat abans del xat, o qualsevol altre que hagués de comunicar diferents clients entre ells.

Hi ha moltes possibles ampliacions pensades per a aquest sistema. La més evident i possiblement la propera, és la realització d'un software client per als controladors aeris, amb les funcionalitats descrites a capítols anteriors.

També seria bo implementar aquests mòduls de comunicació per als principals simuladors de vol, com ara Microsoft Flight Simulator a través del mòdul FSUIPC, necessari per extreure o introduir-ne dades, l'X-Plane, directament mitjançant UDP i amb la documentació que es va crear a l'inici d'aquest projecte sobre la comunicació amb aquest simulador de vol, o el FlightSim, entre d'altres.

7.4. Conclusions personals

La realització d'aquest projecte ha estat molt dura, sobretot a l'última part de la seva elaboració, però a la vegada molt satisfactòria a nivell personal.

M'ha permès dissenyar una aplicació que pot ser d'utilitat a futurs estudiants de l'Escola Politècnica Superior de Castelldefels, participant a l'inici d'un projecte que espero que un pròxim estudiant acabi de realitzar.

Una de les tasques més importants ha estat la definició final dels objectius del projecte, i la seva finalitat.

Per una altra part, m'ha permès conèixer i aprendre un llenguatge de programació que pràcticament no havia vist, com és el Visual Basic, el qual només havia fet servir una vegada a una assignatura i de manera molt superficial.

La fase d'implementació de les aplicacions m'ha permès observar la importància que té la fase de disseny d'un projecte, ja que amb una bon disseny i estudi previ de la situació i dels objectius, la implementació és bastant més ràpida que sense. Una de les aplicacions implementades es va fer sense una bona fase de disseny, i la seva implementació va trigar el doble del que hauria d'haver trigat.

Finalment cal destacar que la realització d'un projecte com aquest és una de les millors maneres de finalitzar la formació d'estudiants d'enginyeria ja que et mostra quines són les necessitats i els mètodes a l'hora de resoldre una sèrie de problemes plantejats i l'assoliment d'objectius.

BIBLIOGRAFIA

[1] Márquez, F. M., *UNIX Programación avanzada*, RA-MA Editorial, Paracuellos de Jarama (Madrid), 1996

[2] Reselman, B., Peasley, R., & Prunchniak, W., *Descubre Visual Basic 6*, Prentice Hall, Madrid, 1999

[3] García de Jalón, J., Rodríguez, J.I., & Brazález, A., *Aprenda Visual Basic 6.0 como si estuviera en primero*, Universidad de Navarra, San Sebastián, 1999

[4] Peña, M., & Cela, J., *Introducción a la programación en C*, Edicions UPC, Barcelona, 2000

[5] "Annex 10 Aeronautical Telecommunications", *Convention on International Civil Aviation*, ICAO, 2000

[6] http://c.conclase.net/librerias/

[7] http://www.geocities.com/chuidiang/#clinux

[8] http://www.programacion.net

[9] http://www.elguille.info/indice.asp

[10] http://labsopa.dis.ulpgc.es/prog_c

[11] http://www.inm.es

[12] http://es.tldp.org/Tutoriales/PROG-SOCKETS/prog-sockets/t1.html

[13] http://msdn.microsoft.com/library/spa/

[14] http://garota.fismat.umich.mx/mn1/manual/manual.html

[15] http://noticias.juridicas.com/base_datos/Admin/rd57-2002.html

[16] http://www.x-plane.info

[17] http://www.jefflewis.net/XPlaneUDP.html

Escola Politècnica Superior de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANNEXOS

TÍTOL DEL TFC: Desenvolupament d'una aplicació de planificació de vol i de comunicació entre simuladors de vol

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica

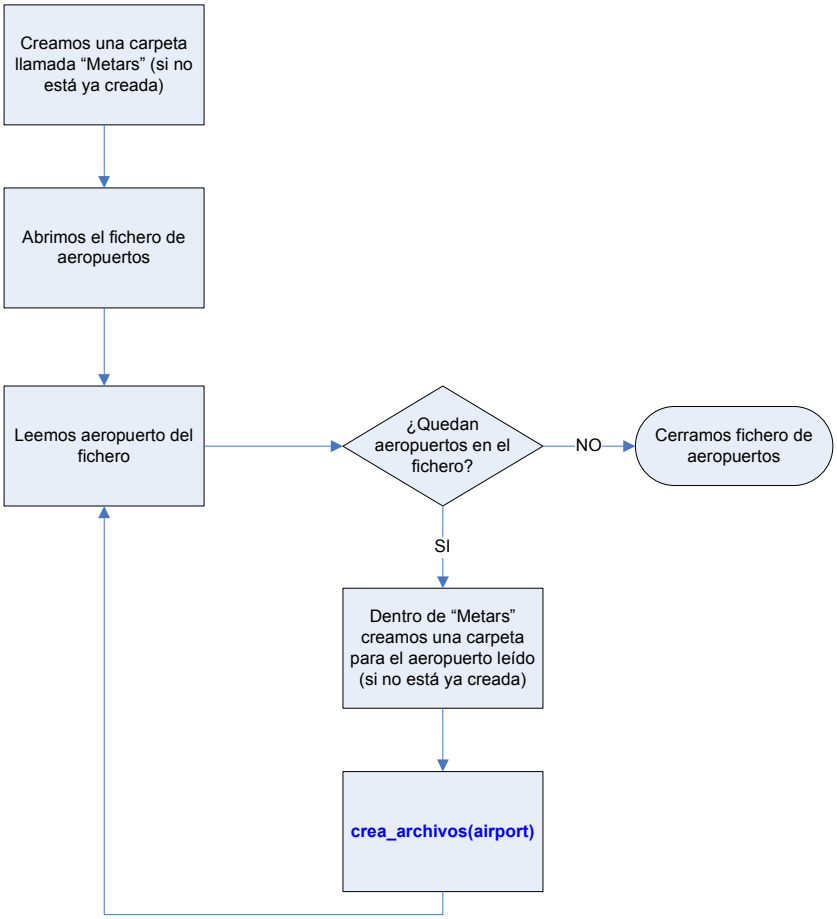
AUTOR: Iván Lorente Díez

DIRECTOR: Xavier Prats i Menéndez

DATA: 24 de febrer de 2005

ANNEX 1: Diagrames de funcionament aplicació METAR

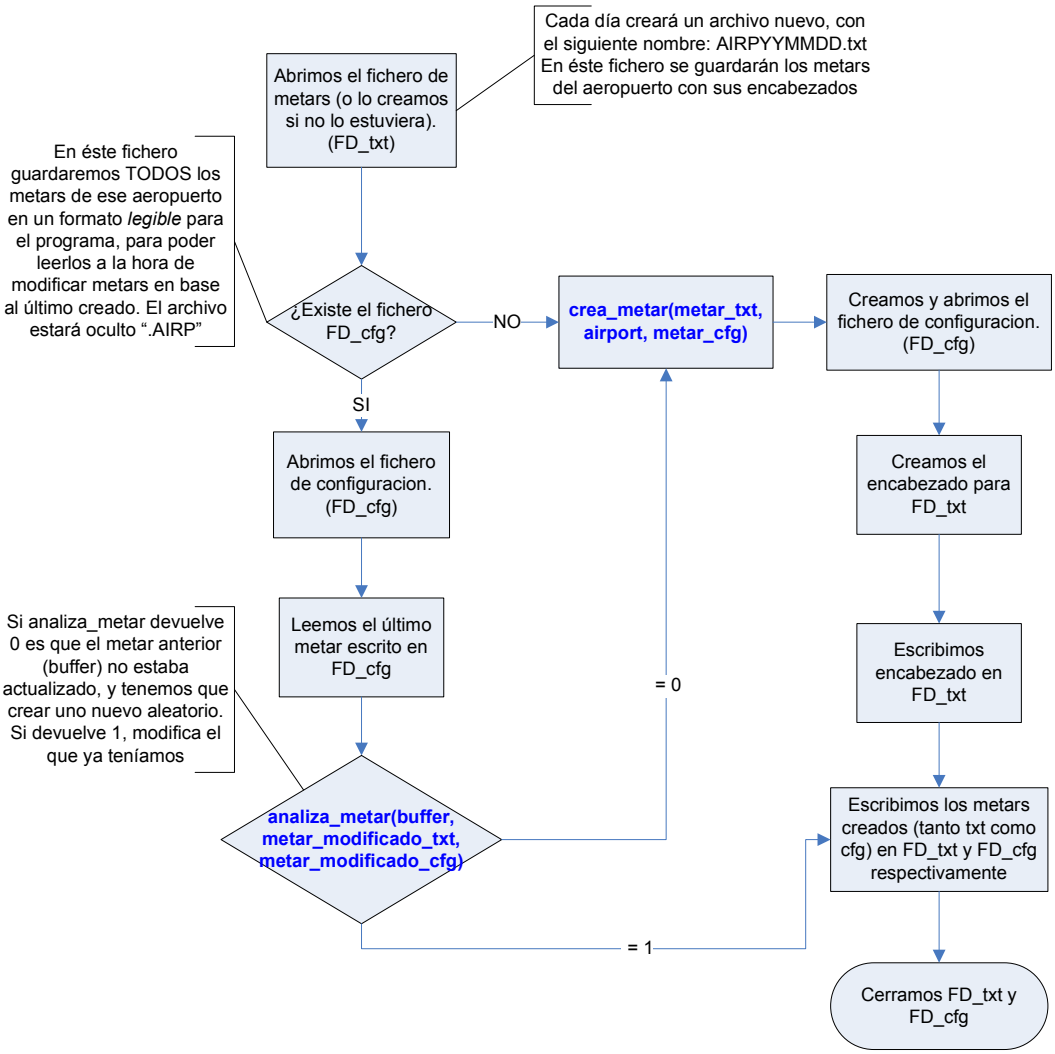
void main (int argc, char *argv[])
{



}

Iván Lorente
2005

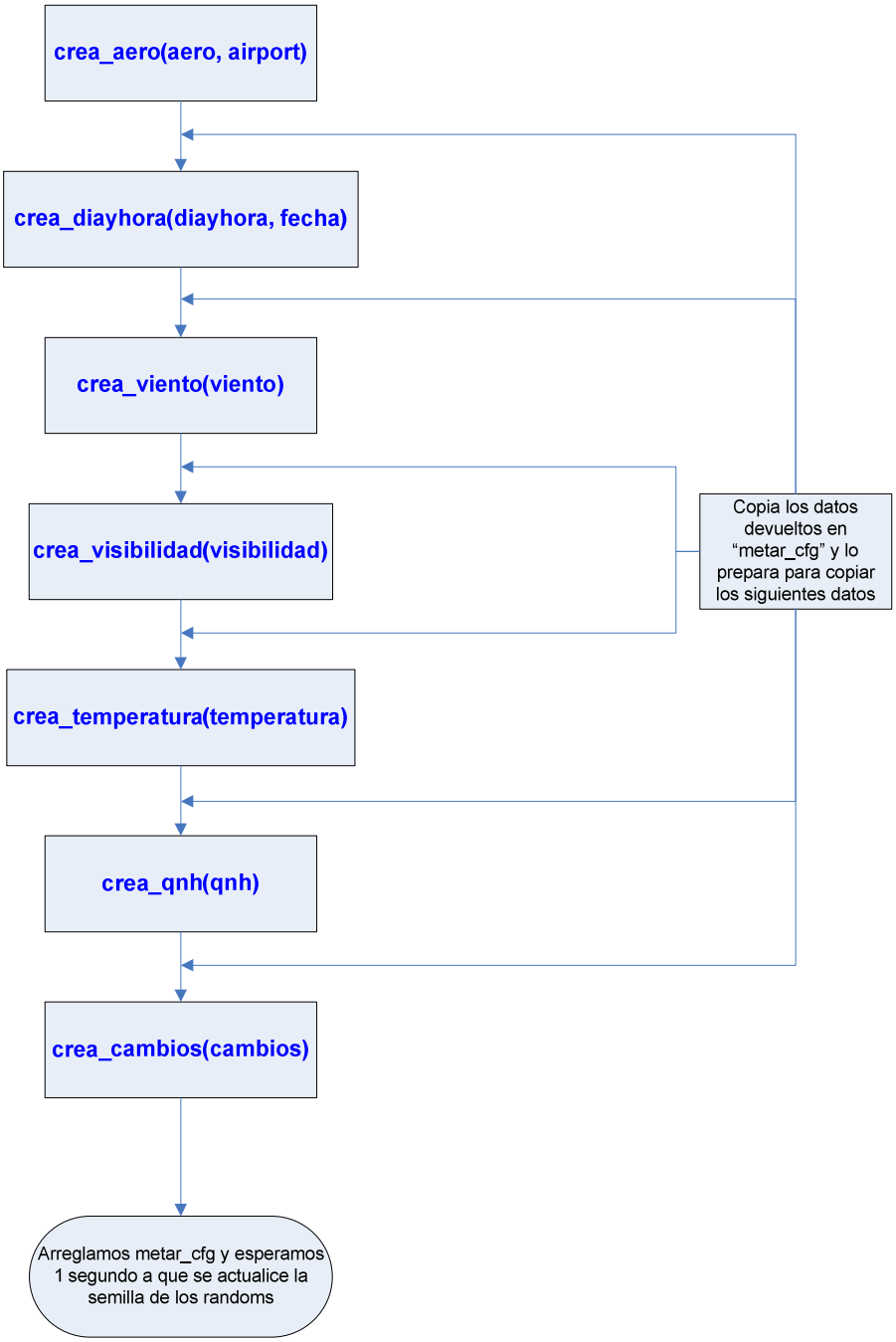
void crea_archivos (char *airport)
{



}

Volver a metar.c

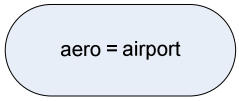
void crea_metar (char *metar, char *airport, char *metar_cfg)
{



}

Volver a crea_archivos

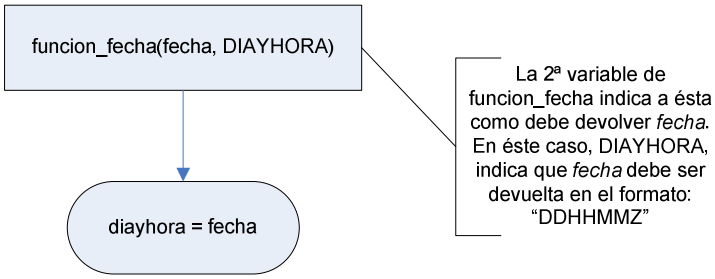
void crea_aero (char *aero, char *airport)
{



}

Volver a crea_metar

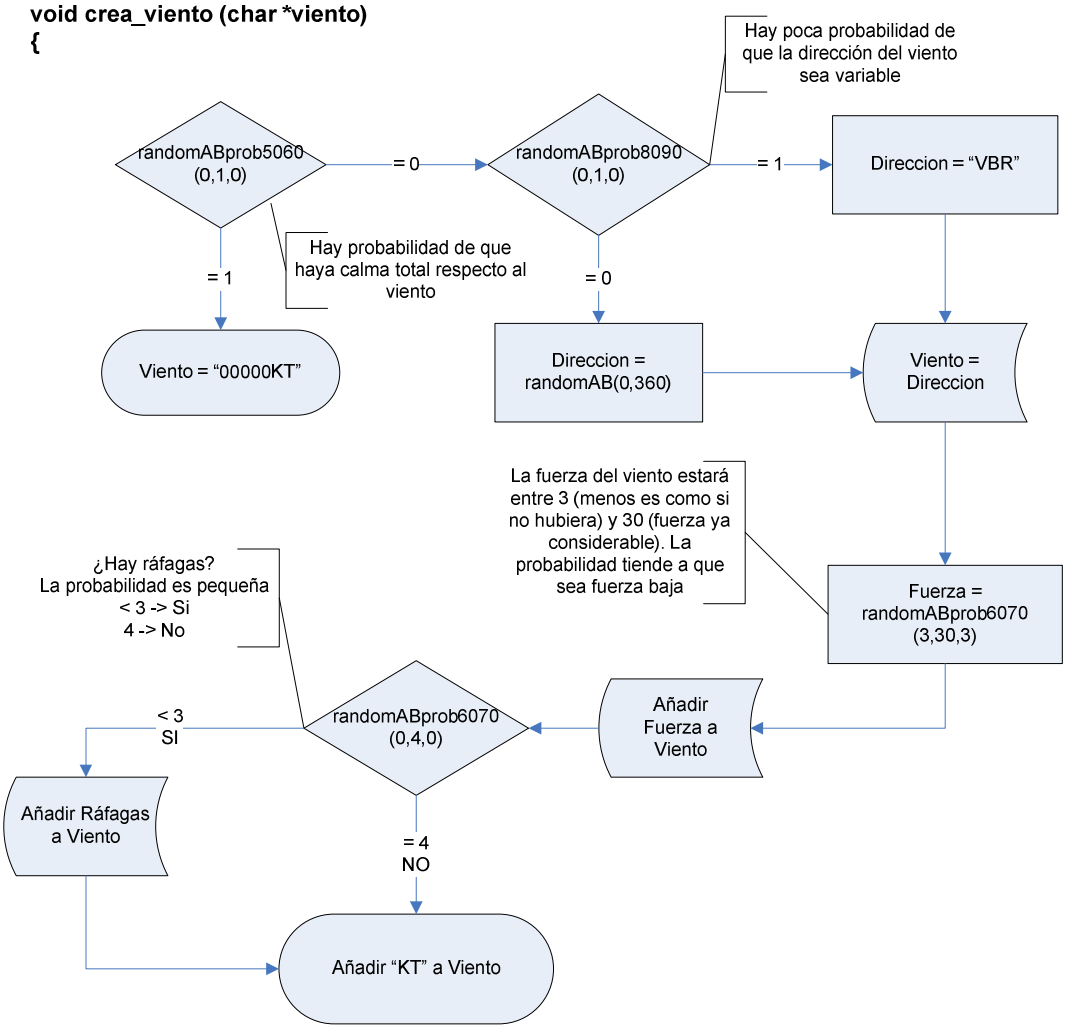
void crea_diayhora (char *diayhora, char *fecha)
{



}

Volver a crea_metar

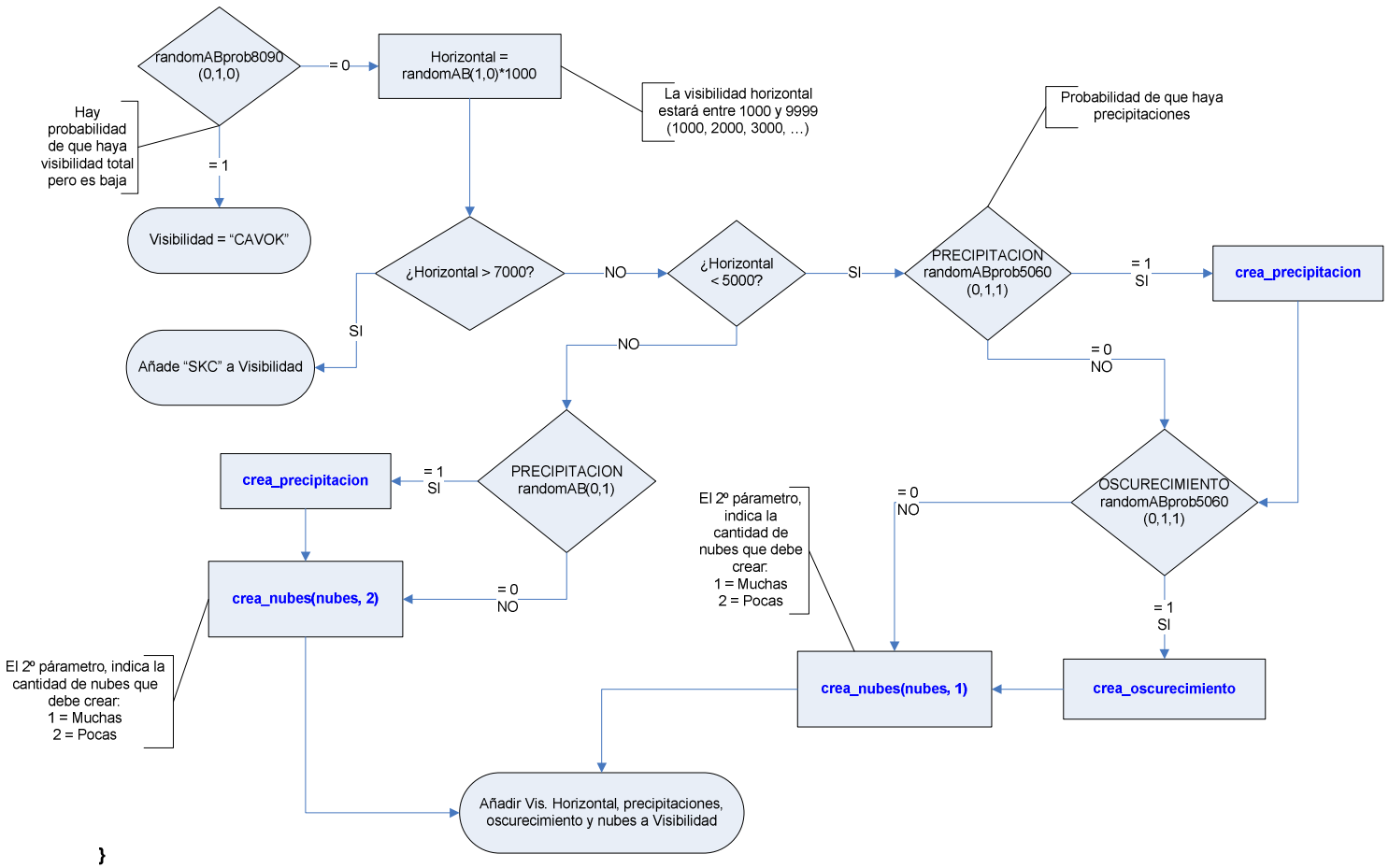
void crea_viento (char *viento)
{



}

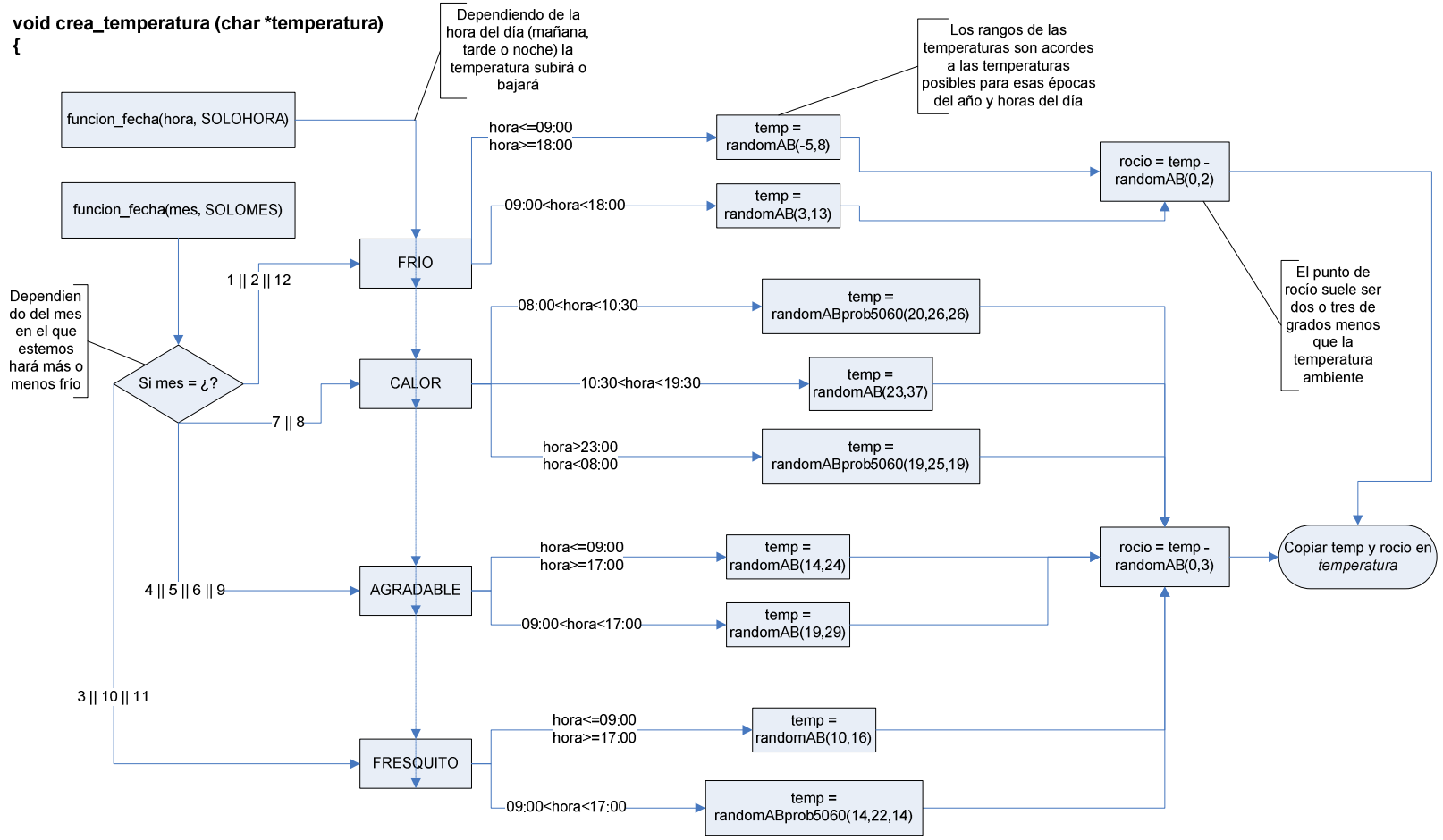
Volver a crea_metar

void crea_visibilidad (char *visibilidad)
{



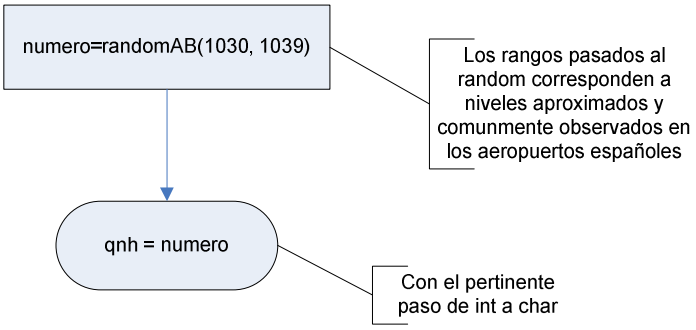
}

Volver a crea_metar



Volver a crea_metar

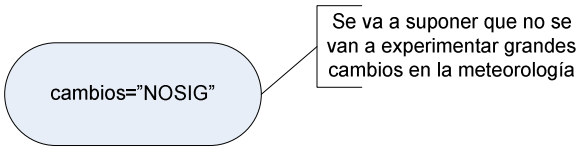
void crea_qnh (char *qnh)
{



}

Volver a crea_metar

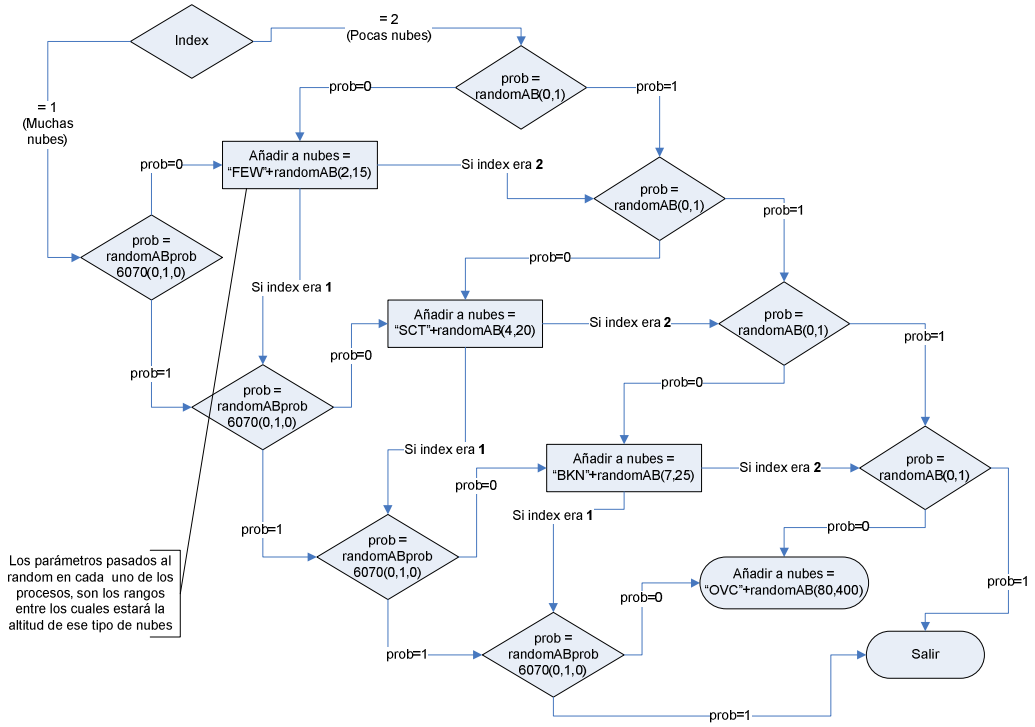
void crea_cambios (char *cambios)
{



}

Volver a crea_metar

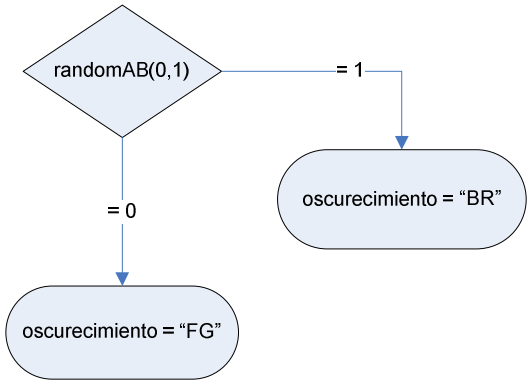
void crea_nubes (char *nubes, int index)
{



}

Volver a crea_visibilidad

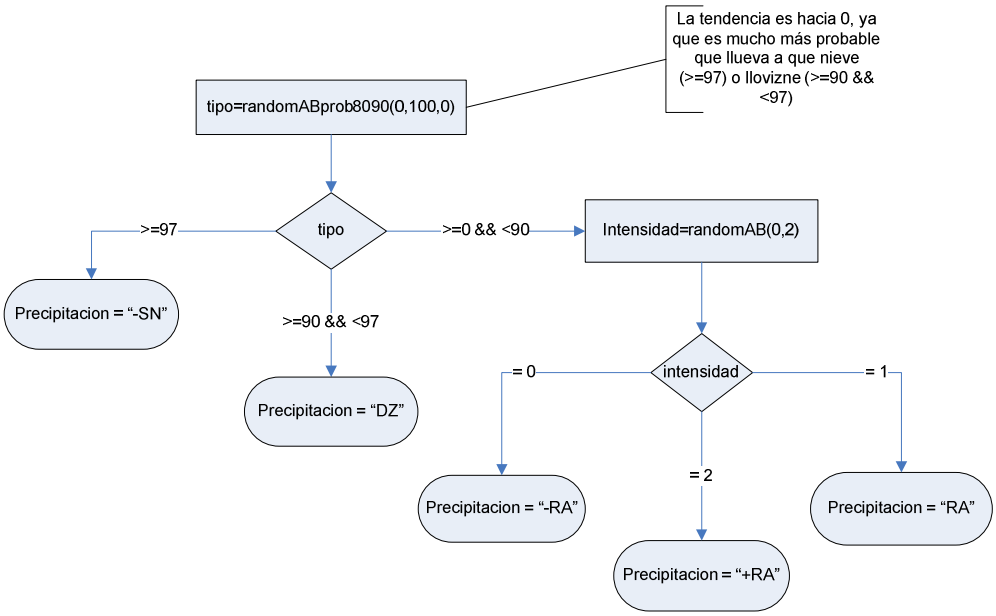
void crea_oscurecimiento (char *oscurecimiento)
{



}

Volver a crea_visibilidad

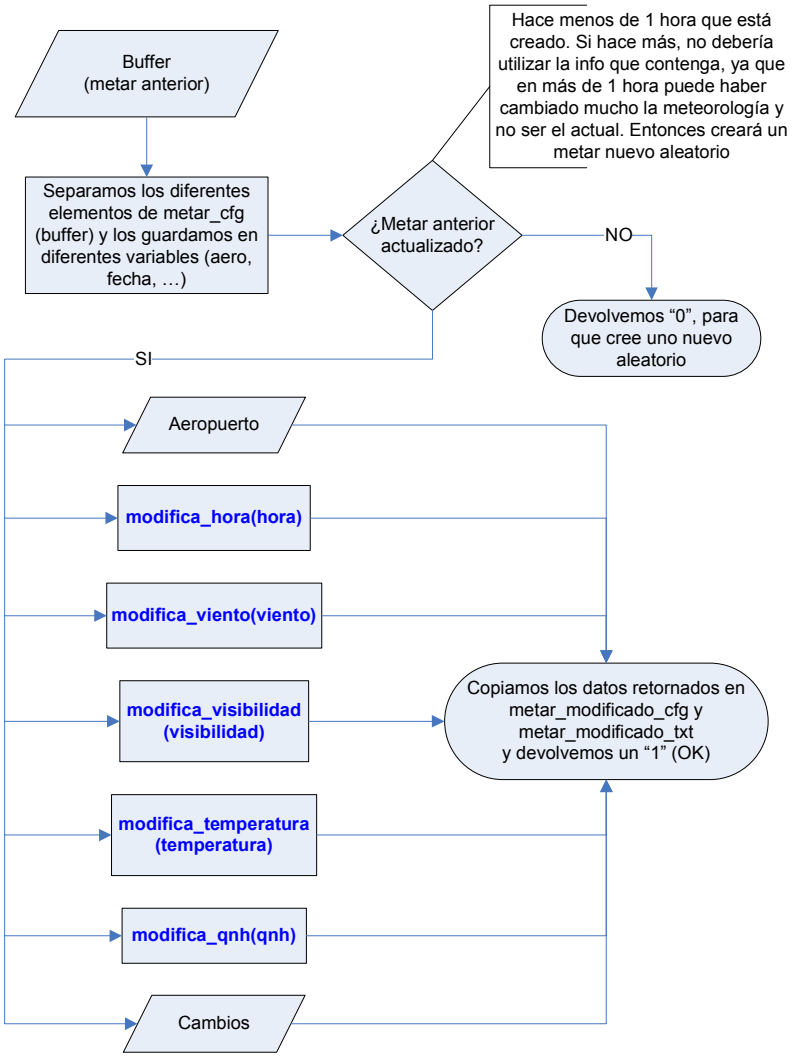
void crea_precipitacion (char *precipitacion)
{



}

Volver a crea_visibilidad

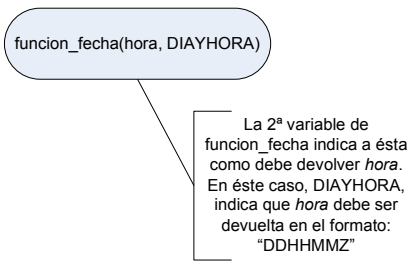
void analiza_metar (char *buffer, char *metar_modificado_txt, char *metar_modificado_cfg)
{



}

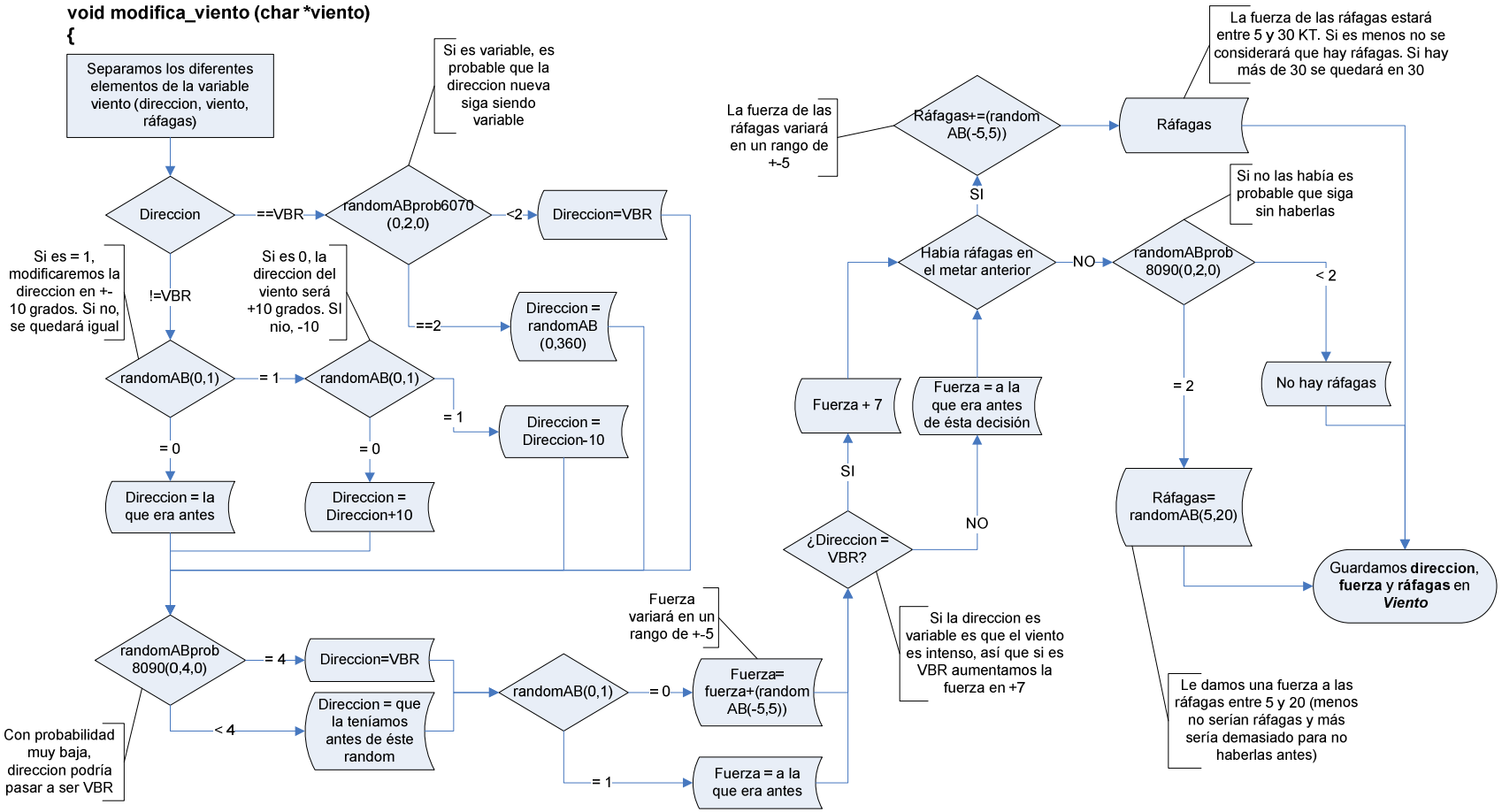
Volver a crea_archivos

void modifica_hora (char *hora)
{



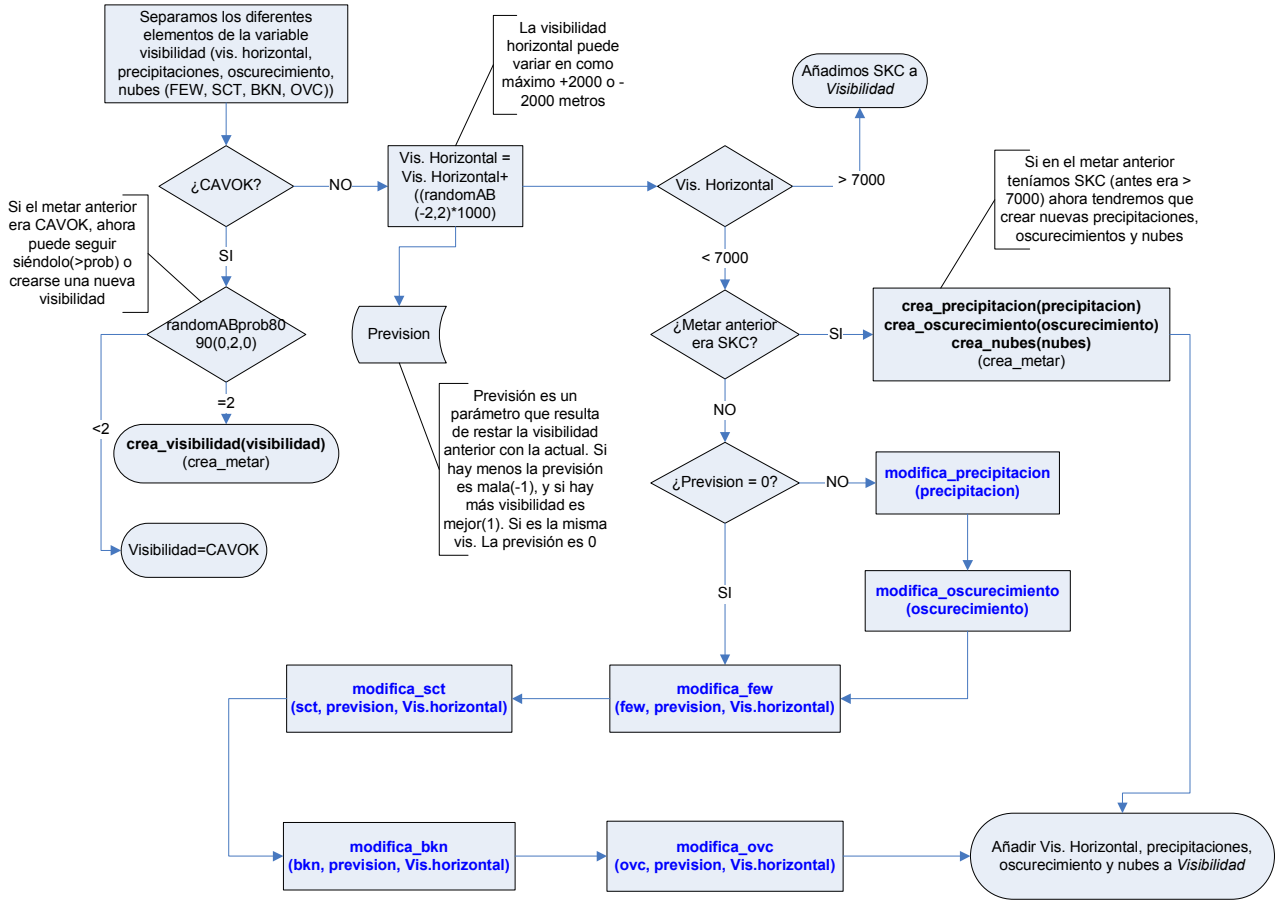
}

Volver a analiza_metar



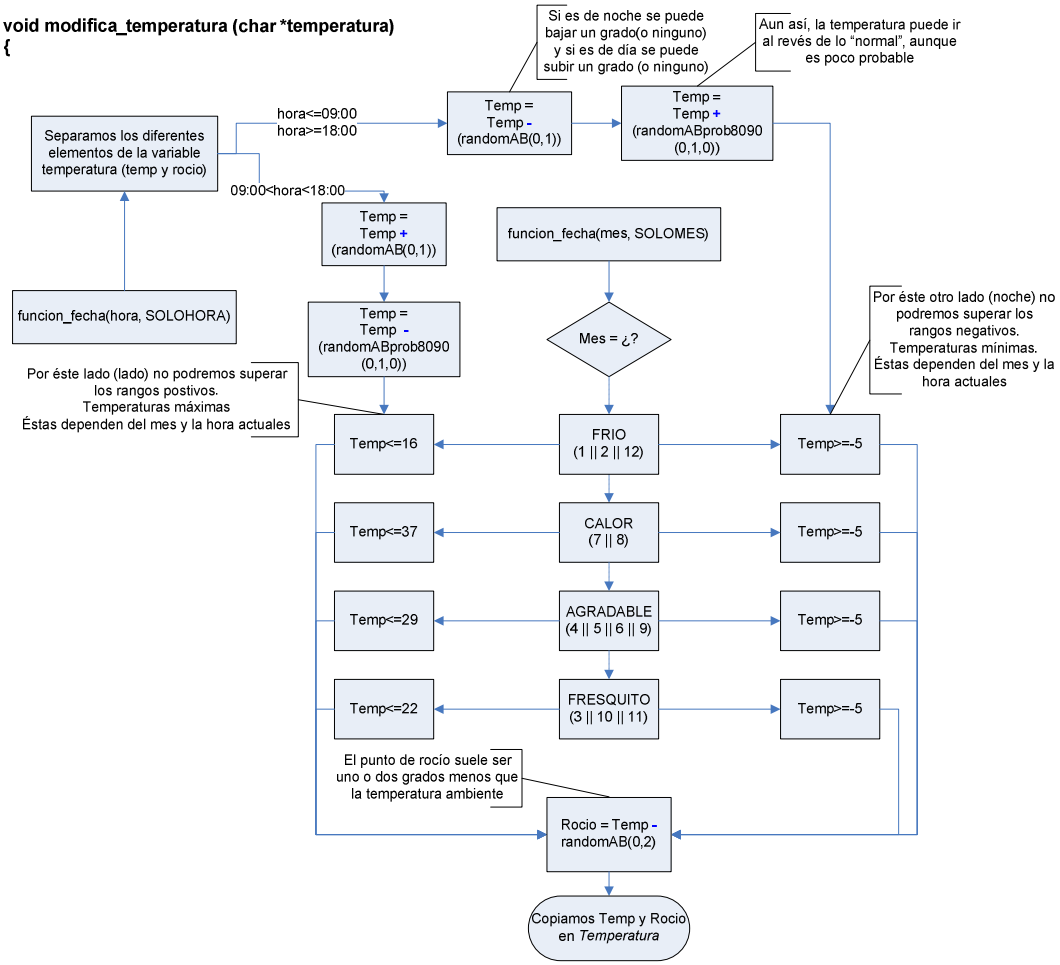
Volver a analiza_metar

void modifica_visibilidad (char *visibilidad)
{

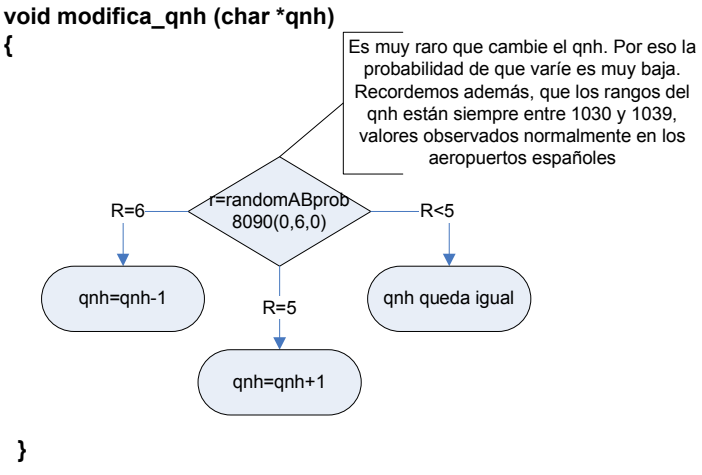


}

Volver a analiza_metar

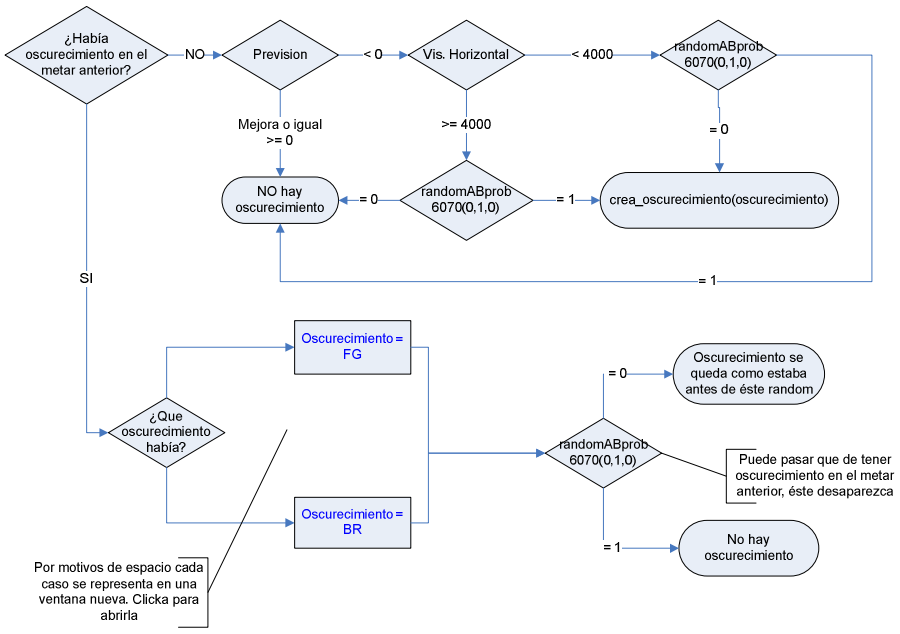


Volver a analiza_metar



Volver a analiza_metar

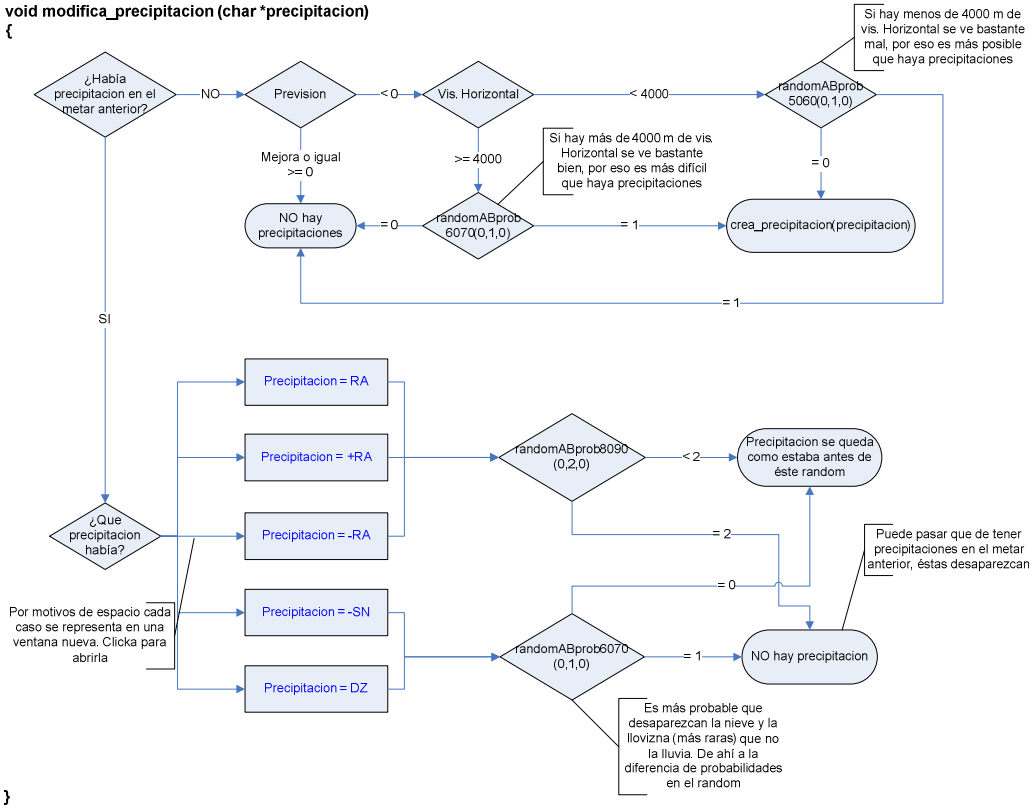
void modifica_oscurecimiento (char *oscurecimiento)
{



}

Volver a modifica_visibilidad

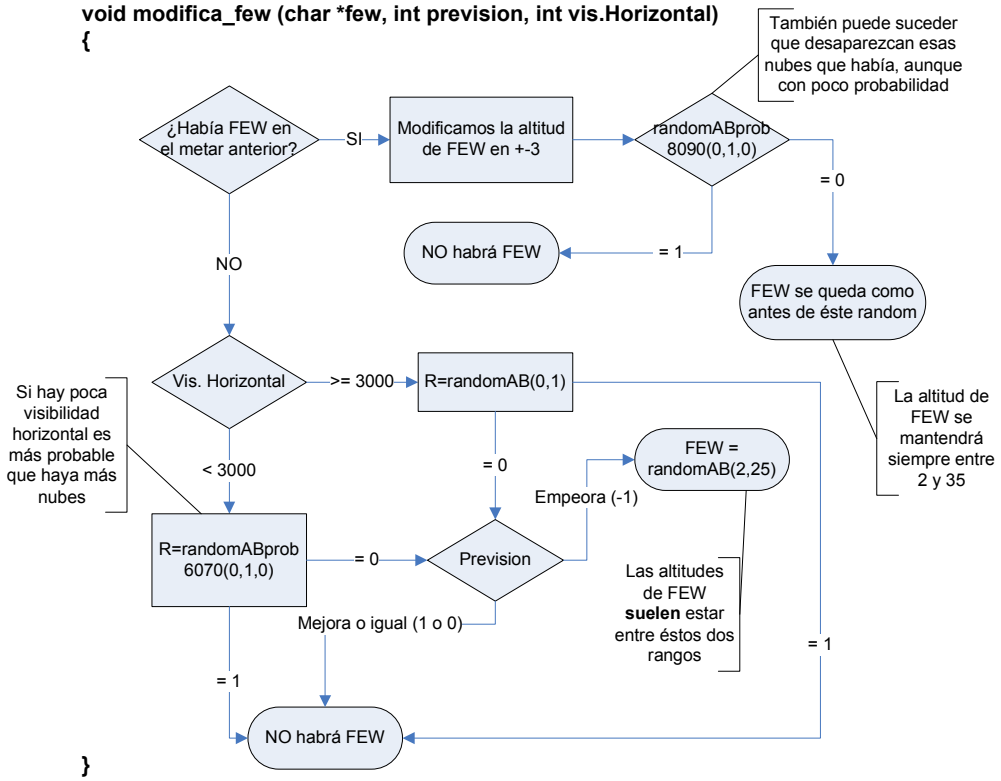
void modifica_precipitacion (char *precipitacion)
{



}

Volver a modifica_visibilidad

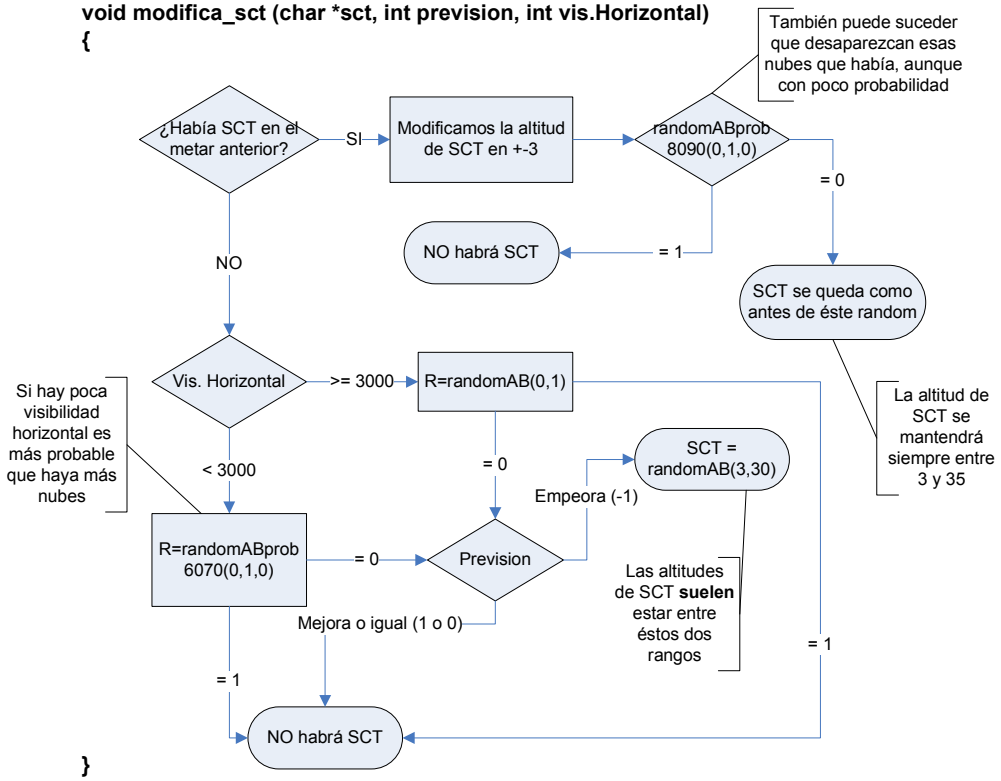
void modifica_few (char *few, int prevision, int vis.Horizontal)
{



}

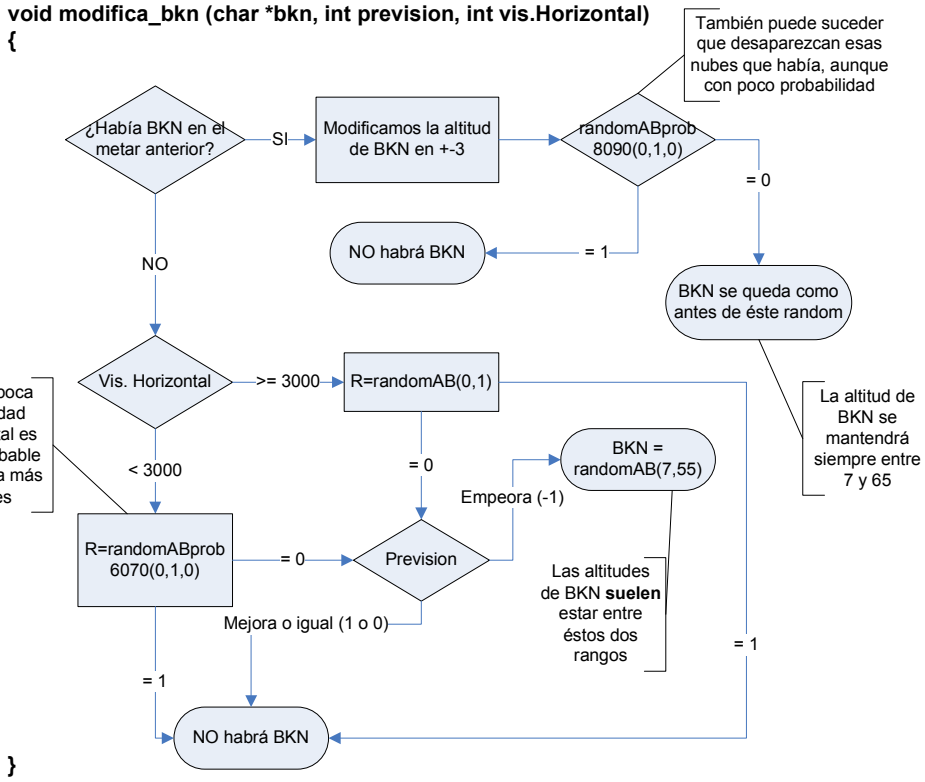
Volver a modifica_visibilidad

void modifica_sct (char *sct, int prevision, int vis.Horizontal)
{

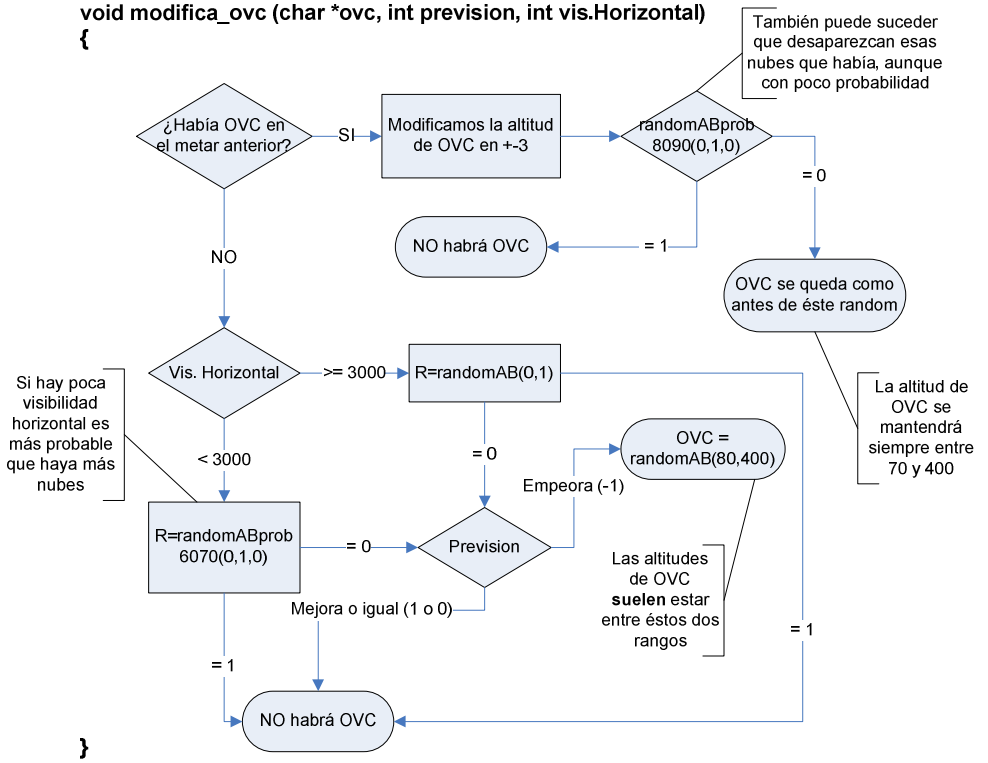


}

Volver a modifica_visibilidad



Volver a modifica_visibilidad



Volver a modifica_visibilidad

ANNEX 2: Índexs X-Plane UDP

Index	Description
0	**Times.**
V0	Frame Rate (frames/sec) - Number of Frames rendered per second. [float]
V1	Time Ratio (ratio) This is how close XP time matches real time. Ideal ratio is 1. [float]
V8	Pausa (0/1) [integer]
1	**Time Elapsed, cockpit Timer...**
V0	Time Elapsed - Seconds. [float]
V1	Cockpit Timer - Seconds. [float]
V2	Local Time (hours expressed digitally) [float]
V3	Zulu Time (hours) [float]
2	**Speed, Vertical Speed**
V0	True Speed in Kts. [float]
V1	Indicated Speed in kts. [float]
V2	True Speed in mph [float]
V3	Indicated Speed in mph [float]
V4	Vertical Speed in feet/minute. [float]
V6	Ground Speed (kts) [float]
V7	Ground Speed (mph) [float]
3	**Mach, G-Loads**
V0	Mach Ratio [float]
V1	G-Load Normal [float]
V2	G-Load Axial [float]
V3	G-Load Side [float]
4	**Atmosphere: Sea level**
V0	Preassure HG
V1	Temperature degC
V2	Wind Speed
V3	Wind Direction
5	**Atmosphere: Ambient**
V0	Preassure HG
V1	Temperature degC
V2	LE temperature
V3	Gravity fts2
V4	Density
V5	Q psf
V6	A ktas
6	**Atmosphere: Systems**
V0	AL preassure HG
V1	Edens
V2	Vacum
7	**Joystick ail/elev/rudd/swe/vec**
V0	Joystick Elevator Input (0-1). (These are the 'user' control inputs) [float]
V1	Joystick Aileron Input (0-1) [float]

V2	Joystick Rudder Input (0-1) [float]
V3	Joystick Sweep Request (Degrees) [float]
V4	Joystick Vector Request (Degrees) [float]
8	**Artificial Stability Values ail/elv/rud**
V0	Elevator. (These are the calculated Artificial Stability Values) [float]
V1	Aileron. [float]
V2	Rudder. [float]
9	**Flight Condition Values ail/elv/rud**
V0	Elevator. (Surface) (These are the sum of UDP #8 and #9) [float]
V1	Aileron (Surface) [float]
V2	Rudder (Surface) [float]
10	**Wing Sweep, Thrust Vector**
V0	Sweep - Ratio [float]
V1	Sweep 1 (deg) [float]
V2	Sweep 2 (deg) [float]
V3	Sweep 3 (deg) [float]
V4	Sweep h (deg) [float]
V5	Thrust Vector (deg) [float]
11	**Trim, Flap, Slat, Speedbrakes**
V0	Elevator Trim [float]
V1	Aileron Trim [float]
V2	Rudder Trim [float]
V3	Rotor Trim [float]
V4	Flap Request
V5	Flap Position
V6	Slat (ratio) [float]
V7	Speedbrake (ratio) [float]
12	**Gears & Brakes**
V0	Gear, 1=Down, 0=Up [integer]
V1	Wheelbrake. Part (1=on)
V2	Left Brake (0-1) [float]
V3	Right Brake (0-1) [float]
13	**Angular Moments**
V0	M (ft/lb) [float]
V1	L (ft/lb) [float]
V2	N (ft/lb) [float]
14	**Angular Accelerations**
V0	Qdot - d/ss [float]
V1	Pdot - d/ss [float]
V2	Rdot - d/ss [float]
15	**Angular Velocities**
V0	Q - d/s [float]
V1	P - d/s [float]
V2	R - d/s [float]
16	**Pitch, Roll, Headings**

V0	Pitch - degrees [float]
V1	Roll - degrees [float]
V2	True Heading - degrees [float]
V3	Magnetic Heading - degrees [float]
V4	Mag Var - degrees [float]
V5	Heading Bug - degrees (true) [float]
17	**Atack angles, paths**
V0	Alpha (angle of attack) - in degrees. [float]
V1	Beta (Sideslip/Yaw) - in degrees. [float]
V2	hpath
V3	vpath
18	**Lat, Lon, Altitude**
V0	Latitude - degrees (Origin Reference Point) [float]
V1	Longitude - degrees [float]
V2	Altitude - fmsl [float]
V3	Altitude - fagl [float]
V5	Altitude - indicated
V6	Latitude - South [float]
V7	Longitude - West [float]
19	**X, Y, Z, distance travelled**
V0	X - m [float]
V1	Y - m [float]
V2	Z - m [float]
V6	Distance - feet [float]
V7	Distance - nm [float]
20	**All Planes: X**
V0	X1 - m [float]
V1	X2 - m [float]
V2	… for all planes
21	**All Planes: Y**
V0	Y1 - m [float]
V1	Y2 - m [float]
V2	… for all planes
22	**All Planes: Z**
V0	Z1 - m [float]
V1	Z2 - m [float]
V2	… for all planes
23	**Throttle Settings**
V0	Throttle Setting (Part). Negative value indicates reverse thrust. [float]
V1	This repeats for each engine...
24	**Engine Settings**
V0	Mode 1
V1	Mode 2
V2	Mode 3
V3	Mode 4

V4	Mode 5
V5	Mode 6
V6	Mode 7
V7	Mode 8
25	**Propellor Settings**
V0	Propellor Setting (rpm). [float]
V1	This repeats for each propellor...
26	**Mixture Settings**
V0	Mixture Setting (Ratio). [float]
V1	This repeats for each Engine...
27	**Carb Heat Settings**
V0	Carb Heat Setting (Ratio). [float]
V1	This repeats for each Carb...
28	**Ignition Settings**
V0	Ignition Switch Position (0,1,2 or 3). [integer]
V1	This repeats for each Ignition...
29	**Cowl Flap Settings**
V0	Cowl Flap Setting (Set). [float]
V1	This repeats for each Cowl...
30	**Engine Power**
V0	Engine 1 - hp [float]
V1	Repeated for each engine...
31	**Engine Thrust**
V0	Engine 1 - lb [float]
V1	Repeated for each engine...
32	**Engine Torque**
V0	Engine 1 - ft/lb [float]
V1	Repeated for each engine...
33	**Engine RPM**
V0	Engine 1 - rpm [float]
V1	Repeats for each engine...
34	**Prop RPM**
V0	Prop 1 - rpm [float]
V1	Repeated for each prop...
35	**Prop Pitch**
V0	Prop 1 - degrees [float]
V1	Repeated for each prop...
36	**Propwash/Jetwash**
V0	Prop 1 - knots [float]
V1	Repeated for each prop...
37	**Manifold Pressures**
V0	Engine 1 - inhg [float]
V1	Repeated for each engine...
38	**N1**
V0	Engine 1 - Percent [float]

V1	Repeated for each engine...
39	**EPR**
V0	Engine 1 - Part [float]
V1	Repeated for each engine...
40	**Fuel Flow**
V0	Engine 1 - lb/hour [float]
V1	Repeated for each engine...
41	**EGT**
V0	Engine 1 - Ratio [float]
V1	Repeated for each engine...
42	**ITT**
V0	ITT1 (deg) [float]
V1	Repeats for each engine...
43	**Oil Pressures**
V0	Pressure 1 (ratio) [float]
V1	Repeats for each engine...
44	**Oil Temperatures**
V0	Temp 1 (ratio) [float]
V1	Repeats for each engine...
45	**Generator amperage**
V0	Amps 1 (ratio) [float]
V1	Repeats for each engine...
46	**Battery Amps**
V0	Battery 1 (ratio) [float]
V1	Repeats for each...
47	**Battery Voltage**
V0	Volts 1 (ratio) [float]
V1	Repeats for each...
48	**Fuel Pumps**
V0	Pump 1 (0/1) [integer]
V1	Repeats for each...
49	**Generators**
V0	Generator 1 (0/1) [integer]
V1	Repeats for each...
50	**Inverters**
V0	Inverter 1 (0/1) [integer]
V1	Repeats for each...
51	**Idle speed**
V0	Idle 1 (0/1) [integer]
V1	Repeats for each...
52	**Starter Timeout**
V0	Starter 1 (secs/1) [float]
V1	Repeats for each...
53	**Payload Weights**
V0	Payload (lb) [float]

V1	Fuel1 (lb) [float]
V2	Fuel2 (lb) [float]
V3	Fuel3 (lb) [float]
V4	Jettison (lb) [float]
V5	Fuel Tank 1 % Full [float] ??? Creo que no hay nada
V6	Fuel Tank 2 % Full [float] ??? Creo que no hay nada
V7	Fuel Tank 3 % Full [float] ??? Creo que no hay nada
54	**Total Weights & CG**
V0	Empty (lb) [float]
V1	Current (lb) [float]
V2	Maximum (lb) [float]
V3	*V3
V4	CofG (ftref) [float]
55	**Aero Forces**
V0	Lift (lb) [float]
V1	Drag (lb) [float]
V2	Side (lb) [float]
56	**Engine Forces**
V0	Normal (lb) [float]
V1	Axial (lb) [float]
V2	Side (lb) [float]
57	**Landing Gear - Vertical Forces**
V0	Gear 1 (lb) [float]
V1	Gear 2 (lb) [float]
V2	Gear 3 (lb) [float]
58	**Landing Gear - Vertical Deflections**
V0	Gear 1 (ft) [float]
V1	Gear 2 (ft) [float]
V2	Gear 3 (ft) [float]
59	**Lift over Drag Ratio**
V0	L/D (Ratio) [float]
60	**Prop Efficiency**
V0	Prop 1 (ratio) [float]
V1	Repeats for each Prop...
61	**Aileron Deflections**
V0	Left Aileron 1 (degrees) [float]
V1	Right Aileron 1 (degrees) [float]
V2	Left Aileron 2 (degrees) [float]
V3	Right Aileron 2 (degrees) [float]
V4	Left Aileron 3 (degrees) [float]
V5	Right Aileron 3 (degrees) [float]
V6	Left Aileron 4 (degrees) [float]
V7	Right Aileron 4 (degrees) [float]
62	**Unknow**
V0	

63	**Roll Spoiler Deflections**
V0	Left Spoiler 1 (degrees) [float]
V1	Right Spoiler 1 (degrees) [float]
V2	Left Spoiler 2 (degrees) [float]
V3	Right Spoiler 2 (degrees) [float]
V4	Left Spoiler 3 (degrees) [float]
V5	Right Spoiler 3 (degrees) [float]
V6	Left Spoiler 4 (degrees) [float]
V7	Right Spoiler 4 (degrees) [float]
64	**Elevator Deflections**
V0	Elevator 1 (degrees) [float]
V1	Elevator 2 (degrees) [float]
65	**Rudder Deflections**
V0	Rudder 1 (degrees) [float]
V1	Rudder 2 (degrees) [float]
66	**Yaw-Brake Deflections**
V0	Left yawb 1 Drudd (degrees) [float]
V1	Right yawb 1 Drudd (degrees) [float]
V2	Left yawb 2 Drudd (degrees) [float]
V3	Right yawb 2 Drudd (degrees) [float]
V4	Left yawb 3 Drudd (degrees) [float]
V5	Right yawb 3 Drudd (degrees) [float]
V6	Left yawb 4 Drudd (degrees) [float]
V7	Right yawb 4 Drudd (degrees) [float]
67	**TOTAL vertical thrust vectors**
V0	Vert1 (vector) [float]
V1	Vert2 (vector) [float]
68	**TOTAL lateral thrust vectors**
V0	Lateral1 (vector) [float]
V1	Lateral2 (vector) [float]
69	**Pitch cyclic Disc Tilts**
V0	cyclic [float]
V1	cyclic [float]
70	**Roll Cyclic Disc Tilts**
V0	cyclic [float]
V1	cyclic [float]
71	**pitch Cyclic Flapping**
V0	flap [float]
V1	flap [float]
72	**Roll Cyclic Flapping**
V0	flap [float]
V1	flap [float]
73	**Ground effect on lift, wings**
V0	Wing1 (L cl*) [float]
V1	Wing1 (R cl*) [float]

V2	Wing2 (L cl*) [float]
V3	Wing2 (R cl*) [float]
V4	Wing3 (L cl*) [float]
V5	Wing3 (R cl*) [float]
V6	Wing4 (L cl*) [float]
V7	Wing4 (R cl*) [float]
74	**Ground effect on drag, wings**
V0	Wing1 (L cd*) [float]
V1	Wing1 (R cd*) [float]
V2	Wing2 (L cd*) [float]
V3	Wing2 (R cd*) [float]
V4	Wing3 (L cd*) [float]
V5	Wing3 (R cd*) [float]
V6	Wing4 (L cd*) [float]
V7	Wing4 (R cd*) [float]
75	**Ground effect on lift, stabs**
V0	Hstab (L cl*) [float]
V1	Hstab (R cl*) [float]
V2	Vstab1 (cl*) [float]
V3	Vstab2 (cl*) [float]
76	**Ground effect on drag, stabs**
V0	Hstab (L cd*) [float]
V1	Hstab (R cd*) [float]
V2	Vstab1 (cd*) [float]
V3	Vstab2 (cd*) [float]
77	**Ground effect on lift, props**
V0	Prop1 (cl*) [float]
V1	Prop2 (cl*) etc [float]
78	**Ground effect on drag, props**
V0	Prop1 (cd*) [float]
V1	Prop2 (cd*) etc [float]
79	**Wing Lift**
V0	Wing1 (lift) [float]
V1	Wing1 (lift) [float]
V2	Wing2 (lift) [float]
V3	Wing2 (lift) [float]
V4	Wing3 (lift) [float]
V5	Wing3 (lift) [float]
V6	Wing4 (lift) [float]
V7	Wing4 (lift) [float]
80	**Wing Drag**
V0	Wing1 (drag) [float]
V1	Wing1 (drag) [float]
V2	Wing2 (drag) [float]
V3	Wing2 (drag) [float]
V4	Wing3 (drag) [float]
V5	Wing3 (drag) [float]
V6	Wing4 (drag) [float]

V7	Wing4 (drag) [float]
81	**Stab Lift**
V0	Hstab (lift) [float]
V1	Hstab (lift) [float]
V2	Vstab1 (lift) [float]
V3	Vstab2 (lift) [float]
82	**Stab Drag**
V0	Hstab (drag) [float]
V1	Hstab (drag) [float]
V2	Vstab1 (drag) [float]
V3	Vstab2 (drag) [float]
83	**Com 1/2 Frequency**
V0	Com 1 Frequency [integer]
V1	Com 1 Standby Frequency [integer]
V2	Com 2 Frequency [integer]
V3	Com 2 Standby Frequency [integer]
84	**NAV 1/2 Frequency**
V0	NAV 1 Frequency [integer]
V1	NAV 1 Standby Frequency [integer]
V2	NAV 2 Frequency [integer]
V3	NAV 2 Standby Frequency [integer]
V4	NAV 1 Type [integer]
V5	NAV 2 Type [integer]
V6	ILS 1 crs [float]
V7	ILS 2 crs [float]
85	**NAV 1/2 OBS**
V0	NAV 1 OBS [float]
V1	NAV 1 flag (0=off, 1=To, 2=From) [integer]
V2	NAV 2 OBS [float]
V3	NAV 2 flag (0=off, 1=To, 2=From) [integer]
86	**NAV 1/2 Deflections**
V0	VOR 1 hdef (degrees) [float]
V1	VOR 1 vdef (degrees) [float]
V2	VOR1 brg [float]
V3	VOR1 distance [float]
V4	VOR 2 hdef (degrees) [float]
V5	VOR 2 vdef (degrees) [float]
V6	VOR2 brg [float]
V7	VOR2 distance [float]
87	**ADF 1/2 Status**
V0	ADF 1 Frequency [integer]
V1	ADF 1 card (degrees) [float]
V2	ADF 1 Relative Bearing (degrees) [float]
V4	ADF 2 Frequency [integer]
V5	ADF 2 card (degrees) [float]
V6	ADF 2 Relative Bearing (degrees) [float]
88	**DME Status**
V0	Select [integer]
V1	Distance [float]
V2	Speed [float]
V3	Time [float]
V4	Found [integer]

V5	NAV01 [integer]
V6	Frequency [integer]
V7	Found [integer]
89	**GPS Status**
V0	Mode (1=APT 2=NDB 3=VOR 11=INT) [integer]
V1	Index [integer]
V2	Distance nm [float]
V3	OBS mag [float]
V4	Course - true [float]
V5	Href dots [float]
V6	Flag (to/from) [integer]
V7	GPS course [float]
90	**Transponder Status**
V0	Frequency [integer]
V1	ID [integer]
V2	Inter (0/1) [integer]
91	**Marker & Audio Morse**
V0	Outer Marker Morse (0/1) [integer]
V1	Middle Marker Morse (0/1) [integer]
V2	Inner Marker Morse (0/1) [integer]
V3	Audio (active) [integer]
V4	Gear Audio (active) [integer]
V5	Lorot Warning [integer]
92	**Switches 1**
V0	Battery (0/1) [integer]
V1	Avionics (0/1) [integer]
V2	Nav Lights (0/1) [integer]
V3	Beacon (0/1) [integer]
V4	Strobe Lights (0/1) [integer]
V5	Landing Lights (0/1) [integer]
V6	HUD Power (0/1) [integer]
V7	HUD Brightness (0-1) [float]
93	**Switches 2**
V0	Pitot Heat (0/1) [integer]
V1	Anti-Ice (0/1) [integer]
V2	Auto-Brake (0/1) [integer]
V3	Taxi light (0/1) [integer]
V4	Yaw Damper (0/1) [integer]
V5	Art Stability (0/1) [integer]
V6	FADEC Power (0-1) [integer]
V7	FADEC Power (0-1) [integer]
94	**Switches 3**
V0	Prop Sync (0/1) [integer]
V1	Arrestor (0/1) [integer]
V2	ECAM (Mode) (0=Engines 1=Fuel 2=Stat (control positions) 3=Hydraulics 4=Failures) [integer]
V3	Radal (Bug) [float]
V4	P-Rot (Power) [integer]
V5	P-Rot (Level) [integer]
V6	Fuel Selector [integer]
V7	Auto Feather [integer]
95	**FS/HSI/RMI/Audio/Map**
V0	Mode [integer]

V1	Pitch (degrees) [float]
V2	Roll (degrees) [float]
V3	HSI (select) [integer]
V4	RMI (select) [integer]
V5	Audio Select. (0=NAV1 1=NAV2 2=ADF1 3=ADF2 5=DME 10=COM1 11=COM2) [integer]
V6	Map Range Select (This is the map range switch position, 0-6) [integer]
96	**Anunciators: General**
V0	Master Caution [integer]
V1	Master Accept [integer]
V2	Auto Disconnect [integer]
V3	Low Vacuum [integer]
V4	Low Voltage [integer]
V5	Fuel Quantity [integer]
V6	Hydraulic Pressure [integer]
V7	Osped test [float]
97	**Anunciators: Engine**
V0	Fuel Pressure [integer]
V1	Oil Pressure [integer]
V2	Oil Temperature [integer]
V3	Inverter Warn [integer]
V4	Generator Warn [integer]
V5	Chip Detect [integer]
V6	Engine Fire [integer]
V7	Auto Ignition [integer]
98	**Autopilot Status**
V0	Speed [integer]
V1	Heading [integer]
V2	Altitude [integer]
V3	Back Course [integer]
V4	Speed [integer]
V5	Heading [integer]
V6	Altitude [float]
V7	VVI [float]
99	**Autopilot Status II**
V0	[integer]
V1	[integer]
V2	[integer]
V3	[integer]
V4	[integer]
V5	[integer]
V6	[float]
V7	[float]
100	**Pressurisation**
V0	Alt (set)
V1	vvi (set)
V2	alt (actual)
V3	vvi (actual)
V4	test (time)
V5	bleed (air)
101	**Weapon Stats**
V0	Heading (to target)
V1	ptch (to target)

V2	R (deg/sec)
V3	Q (deg/sec)
V4	rudder (ratio)
V5	elev (ratio)
V6	V (knots)
V7	Distance (feet)
102	**Warnings**
V0	Annunciator (Test) [integer]
V1	Annunciator (Master Accept) [integer]
V2	Master Caution [integer]
V3	Stall Warning (0/1) [integer]
V4	Ice Warning (0/1) [float]
V5	Paused (0/1) [integer]
103	**Failure States**
V0	Grp 1
V1	Grp 2
V2	Grp 3
V3	Grp 4
V4	Grp 5
V5	Grp 6
V6	Grp 7
V7	Grp 8
104	**Ice States**
V0	Pitot
V1	Prop
V2	Inlet
V3	Windows
V4	Frame
105	**Vspeeds**
V0	Vso (ktas) Stall Speed in Landing Configuration [float]
V1	Vs (ktas) Stall Speed in Cruise Configuration [float]
V2	Vfe (ktas) Max Velocity with Flaps Extended [float]
V3	Vle (ktas) Max Velocity with Gear Extended [float]
V4	Vno (ktas) Max Sturctural Cruising Speed [float]
V5	Vne (ktas) Velocity Never Exceed [float]
V6	Mmo (Mach)
106	**Hardware Options**
V0	Pedal - nobrk [integer]
V1	Pedal - nobrk [integer]
V2	PFC - function [integer]
V3	PFC - Cert [integer]
V4	PFC - Pedal [integer]
V5	PFC - Cirrs [integer]
107	**Camera Location**
V0	X [float]
V1	Y [float]
V2	Z [float]
V3	Pitch (degrees) [float]
V4	Heading (degrees) [float]
V5	Roll (degrees) [float]
V6	Zoom [float]